Instituto Politécnico Nacional
Escuela Superior de Cómputo
TT 2017-B077

Simulator of Logical Operations from a cellular automaton with chaotic behavior to its complex projection

Thesis

To obtain the grade of:
Computer Systems Engineer

Presented by:
Moreno González Gabriela

Director:

Dr. Genaro Juárez Martínez

Mexico City

November 2018

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO
SUBDIRECCIÓN ACADÉMICA
COMISIÓN ACADÉMICA DE TRABAJOS TERMINALES

INSTITUTO POLITÉCNICO NACIONAL
ESCOM

Ciudad de México a 12 de Noviembre de 2018.

LIC. ANDRES ORTIGOZA CAMPOS
PRESIDENTE DE LA COMISIÓN ACADÉMICA
DE TRABAJOS TERMINALES
P R E S E N T E.

Por medio del presente, se informa que la alumna Moreno González Gabriela que integra el equipo del TRABAJO TERMINAL No. 2017-B077, titulado: "Simulador de operaciones lógicas desde un autómata celular con comportamiento caótico a su proyección compleja" concluyó satisfactoriamente su trabajo.

Los discos (DVD's) que se están entregando, fueron revisados ampliamente por su servidor y corregidos, cubriendo el alcance y el objetivo planteados en el protocolo original y de acuerdo a los requisitos establecidos en el Articulo 32 del Documento Rector de Operación y Evaluación para los Trabajos Terminales de la ESCOM.

ATENTAMENTE
"LA TÉCNICA AL SERVICIO DE LA PATRIA"

Genaro Juárez Martínez

NOMBRE Y FIRMA DEL DIRECTOR
DEL TRABAJO TERMINAL

Acknowledgements:

"To my parents
María Claudia González Salinas and Isaac Moreno Avilés,
because both showed me the world in their own way and allow me to see it now to mine."

"To my sisters
Mónica Moreno González and Diana Patricia Moreno González,
because when I needed a friend they were always there."

"For the best teacher in the world
Genaro Juárez Martínez,
for teaching me to love research, and that I can always give more."

"To my friends,
Jesus, Marco and Ulises
for teaching me that there are people for whom it is worth giving everything."

"We can only see little of the future, but enough to realize that there is a lot to do."
Alan Turing

Gabriela Moreno González

Índice general

1. Introduction **21**

2. Fundamental Concepts **23**

2.1. Dynamic Systems 23

2.1.1. Definition of a Dynamic System 23

2.1.2. Classification of Systems 24

2.1.3. Description of the behavior of the Systems 24

2.1.4. Quantitative analysis of a Dynamic System 27

2.2. Cellular Automata 28

2.2.1. Definition 28

2.2.2. Elements that make up a CA 29

2.2.3. Extension of the CA: functions with memory 30

2.3. Elementary Cellular Automata 32

2.3.1. Background 32

2.3.2. Definition 35

2.3.3. Properties 36

2.3.4. Wolfram Classification 37

2.4. Tools used in the analysis of the ECA 37

2.4.1. Lyapunov Exponent for ECA 40

2.4.2. Wuensche Classification 41

2.4.3. The De bruijn Diagrams 42

2.5. Chaotic Systems 44

2.5.1. Definition of a chaotic system 44

2.5.2. Properties 45

2.5.3. Identification of a Chaotic System 46
2.6. Complex Systems 46
2.6.1. Definition of a Complex System 46
2.6.2. Properties 46
2.6.3. Identification of a Complex System 47

3. Rule 126 of ACE 48
3.1. Background 48
3.2. Definition of rule 126 49
3.3. Propiedades 49
3.4. Classification of rule 126 51
3.5. The De Bruijn Diagram for rule 126 51
3.6. Wuensche classification for rule 126 54

4. Rule 126 with memory of 4 generations 57
4.1. Background 57
4.2. Rule 126 with memory: majority function 58
4.3. Properties 58
4.4. Elements within the evolution space 61
4.4.1. Tiles for filtering 61
4.4.2. Gliders 64
4.4.3. Still-life 69
4.4.4. Gliders-gun 70
4.5. Points of collision of the elements within the space of evolutions . . . 72
4.5.1. Collision points of the gliders 73
4.5.2. Collision points of the still-life 77
4.6. Collisions between the elements within the evolution space 79
4.6.1. Binary collisions 80
4.6.2. Ternary collisions 93
4.6.3. Colisiones de orden superior 142

5. Collision-based computing 144
5.1. Background 144
5.2. Definition 147

5.2.1. Main characteristics 147
5.2.2. Objectives 148
5.3. Determining the computing power of a system 148
5.3.1. Chomsky Hierarchy 149
5.3.2. Logic Gates 151

6. Constructions based on collisions 156
6.1. Objects of rule 126 with memory 156
6.1.1. Eaters 156
6.1.2. Black-holes 158
6.1.3. Solitons 159
6.2. Computability of the rule 160
6.2.1. Regular languages 160
6.2.2. Context-Free Grammars 166
6.2.3. Logic gates 169
6.2.4. Rule computing power 177

7. SOL: Logical Operations Simulator 181
7.1. Requirements 181
7.1.1. Functional requirements 181
7.1.2. Non functional requirements 183
7.2. System's architecture 183
7.3. System's modules 184
7.3.1. Elemental Cellular Automatons 184
7.3.2. Rule 126 with memory 187
7.3.3. Logical Operations 190
7.4. Technologies to use 193
7.5. System tests 193
7.5.1. Elemental Cellular Automatons 193
7.5.2. Rule 126 with memory 199
7.5.3. Logical operations 205

8. Final results **209**
8.1. Celebration of late. Prof. Harold V. McIntosh Achievements 2017 . . 209
8.2. The Ninth International Conference on Complex Systems 2018 211

9. Conclusions **213**

10. Future Work **215**

Bibliografía **216**

Glossary **221**

A. Attractors of the rule 126 **222**
A.1. Attractors with length equal to 2 222
A.2. Attractors with length equal to 3 222
A.3. Attractors with length equal to 4 223
A.4. Attractors with length equal to 5 223
A.5. Attractors with length equal to 6 223
A.6. Attractors with length equal to 7 224
A.7. Attractors with length equal to 8 224
A.8. Attractors with length equal to 9 224
A.9. Attractors with length equal to 10 225
A.10. Attractors with length equal to 11 225
A.11. Attractors with length equal to 12 226
A.12. Attractors with length equal to 13 226
A.13. Attractors with length equal to 14 227
A.14. Attractors with length equal to 15 227
A.15. Attractors with length equal to 16 229
A.16. Attractors with length equal to 17 231
A.17. Attractors with length equal to 18 233
A.18. Attractors with length equal to 19 234
A.19. Attractors with length equal to 20 235
A.20. Attractors with length equal to 21 236
A.21. Attractors with length equal to 22 237

A.22. Attractors with length equal to 23 238
A.23. Attractors with length equal to 24 239

B. Collisions made **240**
B.1. Binary collisions 240
B.1.1. Parameters to define the number of binary collisions to be performed 240
B.1.2. Restrictions by dynamics 244
B.1.3. g1 VS g2 244
B.1.4. g1 VS g6 250
B.1.5. g1 VS s1 257
B.1.6. g3 VS g4 260
B.1.7. g3 VS s2 263
B.1.8. g5 VS g2 268
B.1.9. g5 VS g6 274
B.1.10. g5 VS s1 281
B.1.11. s1 VS g2 285
B.1.12. s1 VS g6 288
B.1.13. s2 VS g4 291

Índice de figuras

2.1. Example of a point and limited cycle attractor 25
2.2. Example of a Linear System and a Non-Linear System 26
2.3. Cellular automaton with memory and without memory, applying a local function. 31
2.4. Example of an ECA class I: rule 160. 38
2.5. Graph of the frequency of living cells by evolution: rule 160 38
2.6. Example of an ECA class II: rule 108 38
2.7. Graph of the frequency of living cells by evolution: rule 108 39
2.8. Example of an ECA class III: rule 126 39
2.9. Graph of the frequency of living cells by evolution: rule 126 39
2.10. Example of an ECA class IV: rule 110 40
2.11. Graph of the frequency of living cells by evolution: rule 110 40
2.12. Evolution of rule 126 41
2.13. Evolution of rule 126 with a change in cell 15 41
2.14. Evolution of rule 126 with differences 41
2.15. The De Bruijn diagram corresponding to rule 126 43

3.1. The definition of rule 126 according to its left and right neighbors. . . 49
3.2. Evolution of the rule 126 up to t = 50 with a living cell. 50
3.3. Evolution of the rule 126 up to t = 70 with 30 % of its living cells. . . 50
3.4. Density of the changes in rule 126. 51
3.5. Evolution of the random 126 rule. 52
3.6. Graph of the regular expression. 55
3.7. Graph of the regular expression. 56

4.1. Rule 126 with memory of 4 generations with a central living cell . . . 59
4.2. Rule 126 with memory of 4 generations with a random condition . . . 60
4.3. Tile for filtering 1(fp_1) in the evolutionary space 62
4.4. Tile for filtering 2(fp_2) in the evolutionary space 63
4.5. Glider 1(g_1) moving within the space of evolutions 64
4.6. Glider 2(g_2) moving within the space of evolutions 65
4.7. Glider 3(g_3) moving within the space of evolutions 66
4.8. Glider 4(g_4) moving within the space of evolutions 67
4.9. Glider 5(g_5) moving within the space of evolutions 67
4.10. Glider 6(g_6) moving within the space of evolutions 69
4.11. Still-life (s_1) moving within the space of evolutions 70
4.12. Still-life 2(s_2) moving within the space of evolutions 71
4.13. Glider-gun 1 (gun1) generating the gliders g1 and g2 within the space of evolutions 72
4.14. Glider-gun 2 (gun2) generating the gliders g1 and g2 within the space of evolutions 73
4.15. The collision points of the glider g_1 in the space of evolutions 74
4.16. The collision points of the glider g_2 in the space of evolutions 74
4.17. The collision points of the glider g_3 in the space of evolutions 75
4.18. The collision points of the glider g_4 in the space of evolutions 75
4.19. The collision points of the glider g_5 in the space of evolutions 76
4.20. The collision points of the glider g_6 in the space of evolutions 77
4.21. Collision points of the still-life s_1 in the space of evolutions 77
4.22. Collision points of the still-life s_2 in the space of evolutions 78
4.23. The different results obtained from colliding g_1 and g_2 with its respective equation 80
4.24. The different results obtained from colliding g_1 and g_6 with its respective equation 82
4.25. The collision obtained between g_1 and s_1 83
4.26. The collision obtained between g_3 and g_4 84
4.27. The collision obtained between g_3 and s_2 85
4.28. The collision obtained between g_5 and g_2 86

4.29. The collision obtained between g_5 and g_6 87
4.30. The collision obtained between g_5 and s_1 88
4.31. The collision obtained between s_1 and g_2 89
4.32. The collision obtained between s_1 and g_6 90
4.33. The collision obtained between s_2 and g_4 91
4.34. The different results obtained from colliding $2g_1$ and g_2 with its respective equation 94
4.35. The different results obtained from colliding $2g_1$ and g_6 with its respective equation 95
4.36. The different results obtained from colliding $2g_1$ and s_1 with its respective equation 97
4.37. The different results obtained from colliding $g_1 + g_1$ and g_2 with its respective equation 98
4.38. The different results obtained from colliding $g_1 + g_1$ and g_6 with its respective equation 99
4.39. The different results obtained from colliding $g_1 + g_1$ and s_1 with its respective equation 100
4.40. The different results obtained from colliding g_1 and $2g_2$ with its respective equation 101
4.41. The different results obtained from colliding g_1 and $2s_1$ with its respective equation 102
4.42. The different results obtained from colliding g_1 and $g_2 + g_2$ with its respective equation 103
4.43. The different results obtained from colliding g_1 and g_2^2 with its respective equation 104
4.44. The different results obtained from colliding g_1 and $g_6 + g_4$ with its respective equation 106
4.45. The different results obtained from colliding g_1 and g_6^2 with its respective equation 107
4.46. The different results obtained from colliding g_1 and $g_6 g_4$ with its respective equation 109

4.47. The different results obtained from colliding g_1, s_1 and g_2 with its respective equation . . . 110
4.48. The different results obtained from colliding g_1, s_1 and g_6 with its respective equation . . . 111
4.49. The different results obtained from colliding g_1^2 and g_2 with its respective equation . . . 112
4.50. The different results obtained from colliding g_1^2 and g_6 with its respective equation . . . 113
4.51. The different results obtained from colliding g_1^2 and s_1 with its respective equation . . . 114
4.52. The different results obtained from colliding g_3 and $2s_2$ with its respective equation . . . 115
4.53. The different results obtained from colliding g_3, g_4 and g_2 with its respective equation . . . 116
4.54. The different results obtained from colliding g_3, g_4 and g_6 with its respective equation . . . 117
4.55. The different results obtained from colliding g_3 and g_4^2 with its respective equation . . . 118
4.56. The different results obtained from colliding g_3 and g_4g_2 with its respective equation . . . 118
4.57. The different results obtained from colliding g_3, s_2 and s_2 with its respective equation . . . 119
4.58. The different results obtained from colliding g_3, s_2 and g_4 with its respective equation . . . 120
4.59. The different results obtained from colliding g_3 and s_2^2 with its respective equation . . . 121
4.60. The different results obtained from colliding g_3^2 and g_4 with its respective equation . . . 122
4.61. The different results obtained from colliding g_3^2 and s_2 with its respective equation . . . 123
4.62. The different results obtained from colliding g_5 and $2g_2$ with its respective equation . . . 124

4.63. The different results obtained from colliding g_5 and $2s_1$ with its respective equation . . . 125
4.64. The different results obtained from colliding g_5 and $g_2 + g_2$ with its respective equation . . . 126
4.65. The different results obtained from colliding g_5 and g_2^2 with its respective equation . . . 127
4.66. The different results obtained from colliding g_5 and $g_6 + g_4$ with its respective equation . . . 129
4.67. The different results obtained from colliding g_5 and g_6^2 with its respective equation . . . 130
4.68. The different results obtained from colliding g_5 and $g_6 g_4$ with its respective equation . . . 132
4.69. The different results obtained from colliding g_5, s_1 and g_2 with its respective equation . . . 133
4.70. The different results obtained from colliding g_5 and s_1^2 with its respective equation . . . 133
4.71. The different results obtained from colliding g_5^2 and g_2 with its respective equation . . . 135
4.72. The different results obtained from colliding g_5^2 and g_6 with its respective equation . . . 136
4.73. The different results obtained from colliding g_5^2 and s_1 with its respective equation . . . 137
4.74. Some results of colliding 4 particles within the evolution space . . . 143

5.1. Chomsky Hierarchy . . . 150
5.2. The basic logic gates . . . 155

6.1. Some examples of eaters . . . 157
6.2. An example of a black hole . . . 158
6.3. Some examples of solitons . . . 159
6.4. Non-deterministic Finite automaton of regular language $(01)^*$ . . . 161
6.5. Non-deterministic Finite automaton simplified $(01)^*$ . . . 161

6.6. Non-deterministic Finite Automaton of the regular language $(01)^*$ with the new equivalences . . . 162
6.7. Evaluation of the empty string ϵ . . . 163
6.8. Evaluation of the string 0 . . . 164
6.9. Evaluation of the string 01 . . . 164
6.10. Evaluation of the string 010101010 . . . 165
6.11. Evaluation of the string 01010101 . . . 166
6.12. Generation of the branch tree for the string 010101 . . . 168
6.13. The 4 combinations of the truth table for the XOR gate . . . 170
6.14. The 4 combinations of the truth table for the IF-THEN gate . . . 172
6.15. The 4 combinations of the truth table for the NAND gate . . . 174
6.16. The AND gate built with NAND . . . 175
6.17. The 4 combinations of the truth table for the AND gate . . . 176
6.18. The OR gate built with NAND . . . 177
6.19. The 4 combinations of the truth table for the OR gate . . . 178
6.20. The NOT gate built based on NAND . . . 179
6.21. The 2 combinations of the truth table for the NOT gate . . . 179

7.1. SOL's architecture . . . 183
7.2. SOL's architecture . . . 184
7.3. The classes which make up Elemental Cellular Automatons . . . 185
7.4. The classes which make up Elemental Cellular Automatons . . . 186
7.5. The classes which module of Rule 126 with memory is made of . . . 187
7.6. The described classes which compose the module of Rule 126 with memory . . . 188
7.7. The clsses that compose the Loical Operations'module . . . 191
7.8. The described classes which compose the module of Rule 126 with memory . . . 191
7.9. The system using the Elemental option . . . 194
7.10. The system using the Random setting . . . 196
7.11. The system using the File setting . . . 197
7.12. The system using the Manual option . . . 198
7.13. The evolution using the tile for filtering 1 . . . 200

7.14. The evolution using the tile for filtering 2 201
7.15. 6 gliders and their combinations with tiles for filtering 202
7.16. The 2 still-life and their combinatoins with tiles for filtering 204
7.17. The regular expression's settings 206
7.18. The settings to handle Context-free grammars 207
7.19. The settings to handle logic gates 208

8.1. Some photos of the event Celebration of Late Prof. Harold V. McIntosh Achievements 210
8.2. Some photos of the event The Ninth International Conference on Complex Systems 212

A.1. Attractors with length equal to 16 229
A.2. Attractors with length equal to 17 231
A.3. Attractors with length equal to 18 233
A.4. Attractors with length equal to 19 234
A.5. Attractors with length equal to 20 235
A.6. Attractors with length equal to 21 236
A.7. Attractors with length equal to 22 237
A.8. Attractors with length equal to 23 238
A.9. Attractors with length equal to 24 239

B.1. The 5 different collisions between the glider g_1 in phase 1 and the 5 phases that the glider has g_2 245
B.2. The 5 different collisions between the glider g_1 in phase 2 and the 5 phases that the glider has g_2 246
B.3. The 5 different collisions between the glider g_1 in phase 3 and the 5 phases that the glider has g_2 247
B.4. The 5 different collisions between the glider g_1 in phase 4 and the 5 phases that the glider has g_2 248
B.5. The 5 different collisions between the glider g_1 in phase 5 and the 5 phases that the glider has g_2 249
B.6. The 5 different collisions between the glider g_1 in phase 1 and the 5 phases that the glider has g_6 251

B.7. The 5 different collisions between the glider g_1 in phase 2 and the 5 phases that the glider has g_6 252
B.8. The 5 different collisions between the glider g_1 in phase 3 and the 5 phases that the glider has g_6 254
B.9. The 5 different collisions between the glider g_1 in phase 4 and the 5 phases that the glider has g_6 255
B.10. The 5 different collisions between the glider g_1 in phase 5 and the 5 phases that the glider has g_6 256
B.11. The 5 different collisions between the glider g_1 in phase 1 and the 2 phases that the still-life s_1 has 257
B.12. The 5 different collisions between the glider g_1 in phase 2 and the 2 phases that the still-life s_1 has 258
B.13. The 5 different collisions between the glider g_1 in phase 3 and the 2 phases that the still-life s_1 has 259
B.14. The 5 different collisions between the glider g_1 in phase 4 and the 2 phases that the still-life s_1 has 259
B.15. The 5 different collisions between the glider g_1 in phase 5 and the 2 phases that the still-life s_1 has 260
B.16. The possible collision between the glider g_3 in phase 1 and the glider g_4 in phase 1 261
B.17. The possible collision between the glider g_3 in phase 2 and the glider g_4 in phase 2 262
B.18. The possible collision between the glider g_3 in phase 3 and the glider g_4 in phase 3 262
B.19. The possible collision between the glider g_3 in phase 4 and the glider g_4 in phase 4 263
B.20. The possible collision between the glider g_3 in phase 5 and the glider g_4 in phase 5 264
B.21. The possible collisions between the glider g_3 in phase 1 and the still-life s_2 in phases 1 and 6 265
B.22. The possible collisions between the glider g_3 in phase 2 and the still-life s_2 in phases 2 and 7 265

B.23. The possible collisions between the glider g_3 in phase 3 and the still-life s_2 in phases 3 and 8 266
B.24. The possible collisions between the glider g_3 in phase 4 and the still-life s_2 in phases 4 and 9 267
B.25. The possible collisions between the glider g_3 in phase 5 and the still-life s_2 in phases 5 and 10 267
B.26. The 5 different collisions between the glider g_5 in phase 1 and the 5 phases that the glider g_2 has 269
B.27. The 5 different collisions between the glider g_5 in phase 2 and the 5 phases that the glider g_2 has 270
B.28. The 5 different collisions between the glider g_5 in phase 3 and the 5 phases that the glider g_2 has 271
B.29. The 5 different collisions between the glider g_5 in phase 4 and the 5 phases that the glider g_2 has 272
B.30. The 5 different collisions between the glider g_5 in phase 5 and the 5 phases that the glider g_2 has 273
B.31. The 5 different collisions between the glider g_5 in phase 1 and the 5 phases that the glider g_6 has 275
B.32. The 5 different collisions between the glider g_5 in phase 2 and the 5 phases that the glider g_6 has 276
B.33. The 5 different collisions between the glider g_5 in phase 3 and the 5 phases that the glider g_6 has 278
B.34. The 5 different collisions between the glider g_5 in phase 4 and the 5 phases that the glider g_6 has 279
B.35. The 5 different collisions between the glider g_5 in phase 5 and the 5 phases that the glider g_6 has 280
B.36. The 5 different collisions between the glider g_5 in phase 1 and the 2 phases of the still-life s_1 281
B.37. The 5 different collisions between the glider g_5 in phase 2 and the 2 phases of the still-life s_1 282
B.38. The 5 different collisions between the glider g_5 in phase 3 and the 2 phases of the still-life s_1 283

B.39. The 5 different collisions between the glider g_5 in phase 4 and the 2 phases of the still-life s_1 . . . 284
B.40. The 5 different collisions between the glider g_5 in phase 5 and the 2 phases of the still-life s_1 . . . 284
B.41. The 5 different collisions between the still-life s_1 in phase 1 and the 5 phases that the glider g_2 has . . . 286
B.42. The 5 different collisions between the still-life s_1 in phase 2 and the 5 phases that the glider g_2 has . . . 287
B.43. The 5 different collisions between the still-life s_1 in phase 1 and the 5 phases that the glider g_6 has . . . 289
B.44. The 5 different collisions between the still-life s_1 in phase 2 and the 5 phases that the glider g_6 has . . . 290
B.45. The collision between the still-life s_2 in phase 1 and the glider g_4 in phase 1 . . . 292
B.46. The collision between the still-life s_2 in phase 2 and the glider g_4 in phase 2 . . . 292
B.47. The collision between the still-life s_2 in phase 3 and the glider g_4 in phase 3 . . . 293
B.48. The collision between the still-life s_2 in phase 4 and the glider g_4 in phase 4 . . . 294
B.49. The collision between the still-life s_2 in phase 5 and the glider g_4 in phase 5 . . . 295
B.50. The collision between the still-life s_2 in phase 6 and the glider g_4 in phase 1 . . . 295
B.51. The collision between the still-life s_2 in phase 7 and the glider g_4 in phase 2 . . . 296
B.52. The collision between the still-life s_2 in phase 8 and the glider g_4 in phase 3 . . . 297
B.53. The collision between the still-life s_2 in phase 9 and the glider g_4 in phase 4 . . . 298
B.54. The collision between the still-life s_2 in phase 10 and the glider g_4 in phase 5 . . . 298

Índice de cuadros

4.1. Lines of cells to form the tile for filtering 1 62
4.2. Cell lines to form the tile for filtering 2 64
4.3. Lines of cells to form the glider 1 65
4.4. Lines of cells to form the glider 2 66
4.5. Lines of cells to form the glider 3 67
4.6. Cell lines to form the glider 4 68
4.7. Lines of cells to form the glider 5 68
4.8. Lines of cells to form the glider 6 69
4.9. Cell lines to form the still-life 1 70
4.10. Cell lines to form the still-life 2 72
4.11. Summary table of binary collisions 93
4.12. Summary table of ternary collisions 142

5.1. AND gate truth table 152
5.2. OR gate truth table 152
5.3. NOT gate truth table 152
5.4. NAND gate truth table 153
5.5. NOR gate truth table 153
5.6. XOR gate truth table 154
5.7. XNOR gate truth table 154
5.8. IF gate truth table 155

6.1. Table of transitions of the NFA of the language $(01)^*$ 161
6.2. Table of transitions of the NFA of the language $(01)^*$ with particles . 162
6.3. Truth table of the XOR gate with particles 169

6.4. Truth table of the IF-THEN gate with particles 171
6.5. Truth table of the NAND gate with particles 173
6.6. Truth table of the AND gate which were designed using NAND . . . 173
6.7. Truth table of the OR gate which where designed using NAND 175
6.8. Truth table of the NOT gate with NAND 177

A.1. Attractors for strings of length equal to 2 222
A.2. Attractors for strings of length equal to 3 223
A.3. Attractors for strings of length equal to 4 223
A.4. Attractors for strings of length equal to 5 223
A.5. Attractors for strings of length equal to 6 224
A.6. Attractors for strings of length equal to 7 224
A.7. Attractors for strings of length equal to 8 224
A.8. Attractors for strings of length equal to 9 225
A.9. Attractors for strings of length equal to 10 225
A.10. Attractors for strings of length equal to 11 226
A.11. Attractors for strings of length equal to 12 226
A.12. Attractors for strings of length equal to 13 227
A.13. Attractors for strings of length equal to 14 227
A.14. Attractors for strings of length equal to 15 228
A.15. Attractors for strings of length equal to 16 230
A.16. Attractors for strings of length equal to 17 232

B.1. Table of possible binary collisions 242

Capítulo 1

Introduction

The automata theory is the study of abstract calculation devices, that is, of the " machines ". Alan Turing proposed in the thirties a machine capable of performing the calculations currently performed by computers, his goal was to describe the existing limitations of a calculating machine, these conclusions not only apply to Turing machines, but to all actual real machines.

The word automaton evokes something that pretends to imitate the proper functions of living beings, especially related to movement. In the field of translators, processors, compilers and interpreters, the fundamental thing is not the simulation of the movement, but the simulation of processes to deal with information. The information is encoded in strings of symbols, and an automaton is a device that manipulates chains of symbols that are represented at its entrance, producing other strips or chains at its exit [14].

Thus, the information acts as a raw material that is processed along the automata. For this processing, several models have been defined that allow the manipulation of information according to what one wishes to model, with the emergence of cellular automata.

The cellular automata (AC) are simple mathematical idealizations of natural systems. They are used in the simulation of systems where different elements interact with each other and generate a result according to the space and the states of the elements in the time in which they interact. These mathematical models have proven to generate complex behavior even when the rules that determine the interaction are

simple [39].

The word “ complex ” is a word of the times, like the “ growing complexity of life ” often cited. Science has begun to try to understand complexity in nature, a point contrary to the traditional scientific goal of understanding the fundamental simplicity of the laws of nature. The field of the study of Complex Systems maintains that the dynamics of Complex Systems are based on universal principles that could be used to describe disparate problems from the physics of particles to the economies of societies [6].

In this Thesis 2017-B077 Simulator of Logical Operations from a Cellular Automaton with chaotic behavior to its complex projection, the properties of an AC that define it as a Chaotic System to understand its dynamics and compare it with the complex dynamics presented when adding a function to extend to the AC. The elements that define a complex dynamics called “ particles ” will be described and their interactions with both their environment and other particles will be shown through the modeling of equations. Finally, these equations will be used to define logical gates and thus be able to explore the computational capabilities of CA.

Capítulo 2

Fundamental Concepts

2.1. Dynamic Systems

2.1.1. Definition of a Dynamic System

A Dynamic System is a system that, in reality or conceptually, evolves over time. A simple pendulum is a system that evolves over time. The process of finding a square root using Newton's formula is a system that conceptually evolves over time, but this can also be considered simply as a sequence of approximations.

Dynamic Systems have states denoted by S_i. As it evolves, the System moves through a sequence of states:

$$S_0, ..., S_{i-2}, S_{i-1}, S_i, S_{i+1}, S_{i+2}, ...$$

A state is represented by a value of some kind: an integer, a real number, a vector of real numbers, a matrix, etc. The value should completely represent the state but should not contain redundant information.

The set of all states is called the "state of states" of the system. A sequence of states is called a trajectory or an orbit. When a system evolves from a state S_i to a subsequent state S_{i+1}, we say that you experience a state transition. If we see the system as a direct graph, where the vertices are the states and the transitions are the edges, then a trajectory is a walk in the graph [13].

2.1.2. Classification of Systems

There are different classifications of the Systems according to their input values, their outputs, their behavior, etc. We will only describe according to the input and output data they use, so we find two types of systems:

1. **Discrete Systems:** The discrete systems are those that can be modeled with whole integer values and have points defined in the space to process and obtain an output that is also discrete.

2. **Continuous Systems:** Continuous systems are those that can be modeled with real values, that is, both their inputs and outputs are of ïnfiniteçharacter. They can be seen graphically as a line in space [16].

2.1.3. Description of the behavior of the Systems

Dynamic Systems receive that name due to the behavior they generate as they evolve over time. This behavior has been studied for decades and has generated researchers around the world to propose theorems to understand this behavior. This section will show the ways to describe the behavior that will be used for our system of interest.

Attractors and repellors

The region of the state space in which there are no exit routes is called a Attractor. An attractor that consists of only one state from which its successor is itself is a point attractor (and a balanced state). An attractor that consists of a cycle of states is called a limited cycle. The set of states that eventually lead to an attractor are known as "Basin of attraction"(they are also called as Gardens of Eden [46]) of the attractor.

The opposite of an attractor is a " repellor ", which is a region of the state space in which trajectories arrive but do not enter. A repellent point is a state of unstable equilibrium.

The attractors usually have a lower dimension than the phase space. For example, an attractor could be a line in two dimensions (2D) in a three-dimensional (3D) phase

space. Some attractors have dimensions that are not integers. For example, a row of points may contain too many points to be considered zero-dimensional but not enough points to form a line; said row would have a dimension between 0 and 1. A strange attractor is an attractor with a fractional dimension [13].

In figure 2.1 two attractors are shown, one point and one limited cycle. The blue points symbolize the states that the attractor contains and the red lines symbolize the trajectories that it has. As can be seen, the first one only has a point in which, regardless of the evolutions in time, it will remain infinitely; while the second will also remain cycling but in a finite set of states.

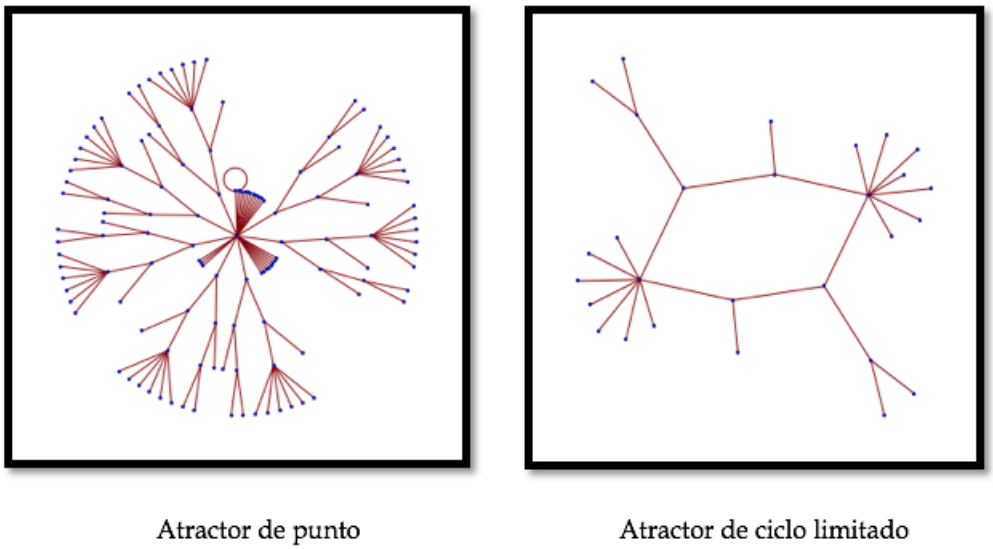


Figura 2.1: Example of a point and limited cycle attractor

Linearity

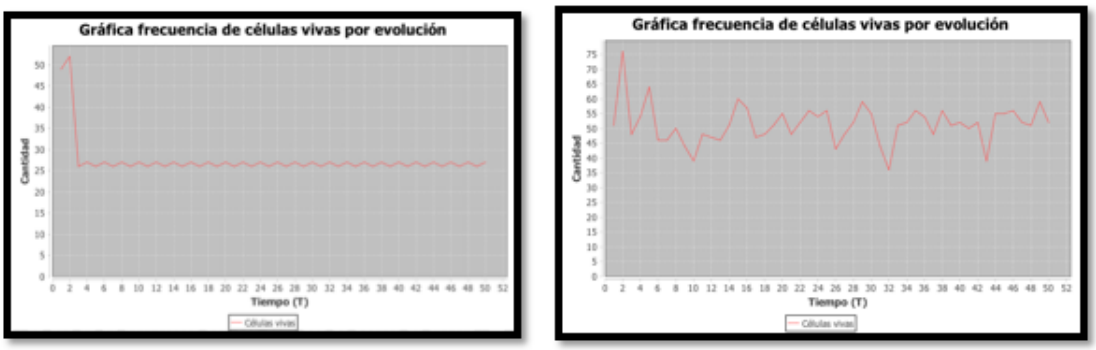
An important property of a very wide class of continuous systems is that, whether linear or non-linear, they behave like linear systems in a small region of phase space that encloses a state of equilibrium. For this reason, much of the classical theory of complex systems focused on their behavior near equilibrium states.

The Linearity implies how much "stableïs a System. It can be observed in a simple way if we imagine a straight line, it maintains a constant shape and it is understandable how it grows as we move through the axes of the catersian plane

(time). However, if we imagine a function that has asymptotes and inflection points, it is no longer trivial to measure how much it grows at what moment of time.

The researchers recently realized that, although they have the convenient property of being easily analyzed, the behavior near equilibrium is not very interesting. This realization led to interest in systems far from equilibrium, and to the study of chaos [16].

In figure 2.2 we can observe two graphs showing the linear and non-linear testing of a System. The red line symbolizes the value of the System in the time shown. The first graph shows how the red line is "maintainedïn a harmonic and simple movement, while in the second graph the red line does not have any distinguishable pattern and it seems that it was generated in a random"way.



Linealidad **No-linealidad**

Figura 2.2: Example of a Linear System and a Non-Linear System

Conditions for chaos

All systems are represented by equations that are usually differential, these equations mathematically model the behavior of the system, contain the inputs, outputs and the relationship between them. Thus, the necessary conditions for a chaotic behavior are:

1. There must be at least three state variables in the system description.

2. At least one of the equations must be non-linear.

2.1.4. Quantitative analysis of a Dynamic System

We already observed in the section 2.1.3 some ways to describe the behavior of Dynamic Systems, however, this is done in a visual way. Now we will see an important component in the analysis of the chaos of a System.

Lyapunov Exponent

This measure of chaos was introduced by the famous Russian mathematician Alexander Mijailovic Lyapunov at the beginning of the 20th century, the Lyapunov Exponents, as they are now known, they are a set of numbers that are usually used to detect the presence of chaos in dynamic systems.

The idea in general is to measure how quickly the contiguous global configurations move away or differ with respect to time.

In chaos theory, there are 3 fundamental properties on chaotic systems(see 2.5 for more information), one of which is sensitive to initial conditions. This is that with very little variation of the initial conditions the result is completely different from our starting point. Thus, we can determine how chaotic a System is according to the variation of the entries [12].

The exponent of Lyapunov for a system is the average value of $\lambda(x)$ for many starting points x. And its calculation is done with the formula of the equation 2.1.4:

$$\lambda(x_0) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \log_e f^{'}(x_i)$$

Where:

1. n is the possible number of states of the System.

2. $f(x_i)$ is the equation that determines the behavior of the System, to this equation we obtain its first derivative.

This exponent allows us to measure the çhaos.ºf our system, however the mathematical calculation is done only for Discrete Systems (2.1.2)

2.2. Cellular Automata

2.2.1. Definition

A Cellular Automata(CA) is defined as a sextuple in the following way:

$$C = (S, S_0, G, d, f, F)$$

Where:

1. S is a finite set of states.
2. S_0 is the initial state, which belongs to S.
3. G_i is a cell's neighborhood of the cell i
4. d is the dimension of the cellular automaton.
5. $f : G \to \Sigma$ is the local function of the automaton, where $G = \{ G_i : i \text{ is a cell}\}$
6. The global mapping, F
   - $F : S \to S$
7. $C(n) = F^{(n)}(S)$ is the state that we get after the iteration n.
8. C(n+1) = F(C(n)) [35]

Also, let's define a cellular automaton as a dynamic system(see 2.1) like:

1. The number of possible states of the cells is finite.
2. The recurrence relation is finite, that is, the new state of a cell depends solely on the state of a finite number of cells in the previous instant.
3. The recurrence relation is invariant in the transfer of space and time, that is, it does not make use of the absolute position of a cell in space and time [3].

2.2.2. Elements that make up a CA

The cellular automata have basic elements, these are:

1. **Regular Fix:** Whether it is a 1-dimensional plane or an n-dimensional space, this is the space of evolutions, and each homogeneous division of the array is called a cell.

2. **Initial setup:** It consists of assigning a status to each of the cells in the initial evolution space of the system

3. **Neighborhoods:** Defines the contiguous set of cells and relative position with respect to each of them. To each different neighborhood corresponds an element of the set of states.

4. **Local Function:** It is the evolution rule that determines the behavior of the AC. It is made up of a central cell and its neighborhoods. It defines how each cell must change state depending on the previous states of its neighborhoods. It can be an algebraic expression [20].

Additionally, in order to better understand its visual representation, it is necessary to mention the types of boundaries or borders, of the plane in which it is developed, in which it is classified:

1. **Open Border:** It is considered that all the cells outside the space of the automaton take a fixed value.

2. **Reflective border:** The cells outside the space of the automaton take the values that are inside, as if it were a mirror.

3. **Periodic or Circular Border:** The cells that are on the border interact with their immediate neighbors and with the cells that are at the opposite end of the array, as if we were folding the plane like a cylinder.

4. **Without Border:** The representation of automaton has no limits, it is infinite. This is only practical when you have software that simulates the evolution of the automaton [11].

2.2.3. Extension of the CA: functions with memory

The conventional cellular automata are ahistorical (without memory), the new state of a cell depends on the configuration of the neighbors only in the previous step in the time of φ. The Cellular Automata with memory can be considered as an extension of the standard reference frame of cellular automata where each cell x_i he is allowed to remember some period of his previous evolutions. Basically, memory is based on the state and history of the system, so we design a memory function φ as follows:

$\varphi(X_i^{t-\tau}, ..., X_i^{t-1}, X_i^t) \longrightarrow S_i$

such that $\tau < t$ determines the degree of backward memory and each cell $displaystyleS_i\epsilon\Sigma$ is a function of the series of states in the cell $displaystyleX_i$ until the passage of time $displaystylet - \tau$. Finally, to execute the evolution, we apply the original rule as follows:

$\varphi(..., S_{i-1}^t, S_i^t, S_{i+1}^t, ...) \longrightarrow X_i^{t+1}$

In cellular automata with memory, while the mapping φ remains unchanged, a historical memory of past iterations is conserved by each cell as a summary of its previous states; therefore, all cells map memory to the mapping φ. As an example, we can take the memory function φ as a majority memory:

$\varphi_{maj} \longrightarrow S_i$

Then, φ_{maj} represents the classical majority function for three variables, as follows:

$\varphi_{maj} : (X_1 \wedge X_2) \vee (X_2 \wedge X_3) \vee (X_3 \wedge X_1) \longrightarrow X$

in the cells $(X_i^{t-\tau}, ..., X_i^{t-1}, X_i^t)$ and define a temporary ring before calculating the following global configuration c. In case of a loop, the majority function allows breaking it in favor of zero if $X_{\tau-1} = 0$, or for one if $X_{\tau-1} = 1$. The representation of an elementary cellular automaton with memory is given as follows:

$\varphi_{CARm} : \tau$

Where CAR represents the decimal notation of a particular elementary cellular automaton and m the type of memory given with a specific value of τ. Thus, the majority memory (maj) working on the 126th rule of the elementary cellular automata checking 4 cells ($\tau = 4$) of history is denoted simply as $R126_{maj} : 3$. The figura 2.3 represents in detail the memory working on an elementary cellular automaton.

Memory is as simple as any cellular automaton and its local function but sometimes the global behavior produced by the local rule is totally unpredictable. In the figure 2.3 we can observe how memory works in comparison with an ahistorical automaton.

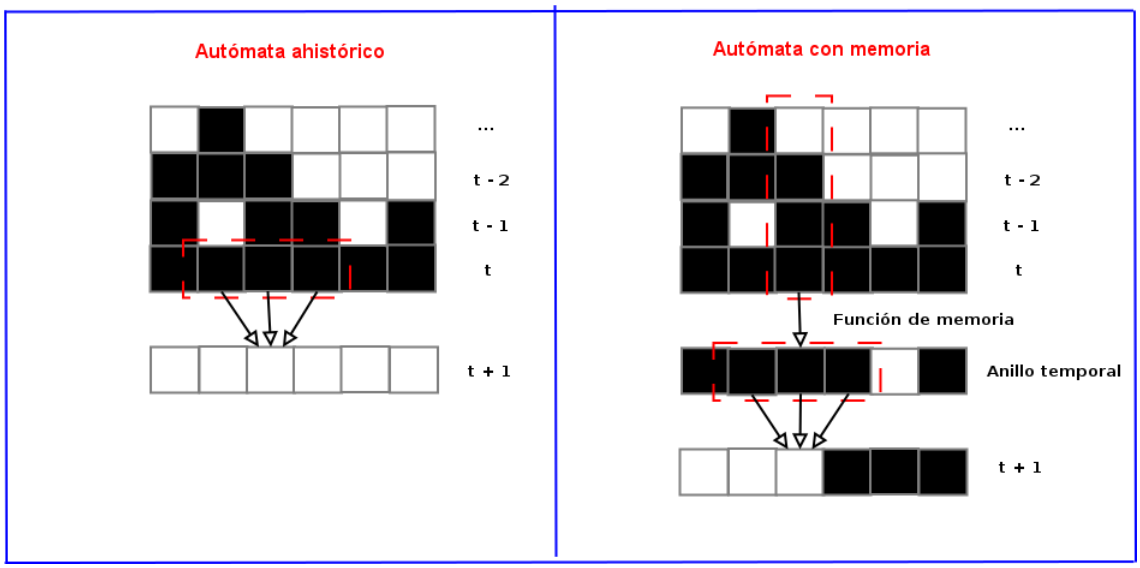


Figura 2.3: Cellular automaton with memory and without memory, applying a local function.

Some authors define memory rules as those with dependence on φ in the state of the cell to be updated. Then the rules of a dimension without memory take the form: $X^{t+1} = \varphi(X^t_{i-1}, X^t_{i+1})$. The use of associative locution memory usually refers, when used in the context of cellular automata, to the study of the configurations of attractors, which are argued by Wuensche to constitute the global contents of the addressable memory network in the meaning of Hopfield. This study of the attractors will be discussed in detail in chapter 2.3.2.

There are several functions that allow you to manage the part with memory, which are:

1. Majority (majority): This function starts from the number of evolutions passed to be taken. Imagine that we are going to take up $\tau = t-4$, there are 3 possible situations: if the number of living cells is greater than that of dead cells, a 1 will be added to the temporary ring; If the number of dead cells is greater than

that of living cells, then a 0 will be added to the time ring, if it is the same number of living cells as dead cells, the value of $\tau = t - 1$.

2. Minority (minority): Taking up again $\tau = t - 4$, there are 3 possible situations: if the number of living cells is less than that of dead cells, a 1 will be added to the temporary ring; if the number of dead cells is less than that of living cells, a 0 will be added to the temporary ring, if the same number is added, the value of $\tau = t - 1$

3. Parity(parity): This function implies that if we take up $\tau = t - 4$ then the values of the 4 cells will be added(zeros and ones), and to that value a modulus 2 will be applied, that is, if the value of the sum is odd then the value to add in the temporal ring is a 1, unless, a 0 will be added.

In all cases, when you have the temporary ring the elementary rule will be applied to it and the result will be the next evolution of the automaton.

2.3. Elementary Cellular Automata

2.3.1. Background

Despite its very simple construction, nothing like the general cellular automata seems to have been considered before around the fifties. However, in the 1950s - inspired in various ways by the advent of electronic computers - several different types of systems equivalent to cellular automata were introduced independently. A variety of precursors can be identified. Operations on sequences of digits had been used since antiquity in arithmetic. The approximations of finite differences to differential equations began to emerge in the early twentieth century and were well known in the thirties. And the Turing machines invented in 1936 were based on the thought of arbitrary operations on sequences of discrete elements.

The most familiar form in which cellular automata were introduced (and which eventually led to their name) was through the work of John von Neumann in the attempt to develop an abstract model of self-reproduction in biology-a subject that had emerged of the investigations in cybernetics. Around 1947, perhaps based on

chemical engineering, von Neumann began thinking about models based on 3D factories described by partial differential equations. He soon switched to thinking about robotics and imagined maybe implementing an example using a toy building set. By analogy with electronic circuit designs, he realized that 2D should be enough. And following a 1951 suggestion by Stanislaw Ulam (which may have already independently considered the problem) he simplified his model and ended up with a 2D cellular automaton (he apparently later hoped to convert the results back to differential equations). The particular cellular automaton he built in 1952 had 29 possible colors for each cell and complicated rules specifically configured to emulate the operations of electronic computer components and various mechanical devices. To give a mathematical proof of the possibility of self-reproduction, von Neumann then described the construction of a cellular configuration of 200,000 to be reproduced (the details were filled in by Arthur Burks in the early 1960s). Von Neumann seems to have believed, presumably in part, by seeing the complexity of real biological organisms and electronic computers, that something like this level of complexity would inevitably be necessary for a system to exhibit sophisticated capabilities such as self-reproduction.

By the end of the 1950s it had been observed that cellular automata could be seen as parallel computers, and particularly in the 1960s a sequence of increasingly detailed and technical theorems was demonstrated - often analogous to those of Turing machines - about their formal computational abilities. Attempts at connecting the cellular automata to the mathematical discussions of dynamic systems began in the late 1960s, although, as discussed below, this had already been done a decade earlier, with different terminology. And by the mid-1970s work on cellular automata had become quite esoteric and interest in it had largely vanished. (However, some work continued, especially in Russia and Japan). Note that even in computer science, various names were used for cellular automata, including tessellation automata, cellular spaces, iterative automata, homogeneous structures, and universal spaces.

Despite the lack of scientific research, an example of a cellular automaton entered the recreational computing in an important way in the early seventies. Apparently motivated in part by issues of mathematical logic, and partly by the work of " simulation games " by Ulam and others, John Conway in 1968 started doing experiments

(mostly by hand, but later on a PDP computer). -7) with a variety of different rules of 2D cellular automata, and in 1970 had developed a simple set of rules that he called " The Game of Life ", which exhibit a series of complex behaviors. Largely through popularization in Scientific American by Martin Gardner, life became widely known. An immense amount of effort was devoted to finding initial special conditions that give particular forms of repetitive or other behavior, but virtually no systematic scientific work was done (perhaps in part because even Conway treated the system largely as a recreation) and almost without exception only The very specific rules of Life were investigated. (In 1978 as a possible 1D analogue of life easier to implement in the first personal computers, Jonathan Millen nevertheless briefly considered what turns out to be the code 20 k = 2, r = 2 totalist rule.)

Quite disconnected from all this, even in the 1950s, specific types of 2D and 1D cellular automata were already being used in various electronic devices and special purpose computers. In fact, when digital image processing began in the mid-fifties (for applications such as optical character recognition and microscopic particle counting), the rules of 2D cellular automata used to be used to eliminate noise. And for several decades, starting in 1960, a long line of so-called cellular logic systems was built to implement 2D cellular automata, mainly for image processing. Most of the rules used were established specifically to have a simple behavior, but occasionally it was observed as a largely recreational issue that, for example, patterns of alternating stripes (" custering ") could be generated.

In pure mathematics, infinite sequences of 0 and 1 have been considered in various forms since at least the 1800s. Beginning in the 1930s, the development of symbolic dynamics led to the investigation of the assignments of such sequences to whether same. And in the mid-fifties studies were being made (especially by Gustav Hedlund) of the so-called switch-commutation block maps, which happen to be exactly 1D cellular automata. In the fifties and early sixties there was work in this area (at least in the United States) by distinguished pure mathematicians, but as it was largely for application to cryptography, much of it was kept secret . And what was published was mostly abstract theorems about characteristics too global to reveal any kind of complexity that was discussed.

Specific types of cellular automata have also emerged - usually under different

names - in a wide range of situations. In the late fifties and early sixties, 1D cellular automata were studied as a way to optimize circuits for arithmetic and other operations. Starting in the 1960s, simulations of idealized neural networks sometimes had neurons connected to neighbors in a grid, producing a 2D cellular automaton. Similarly, several models of active media - particularly heart and other muscles - and reaction diffusion processes used a discrete grid and discrete excitation states, corresponding to a 2D cellular automaton. (In physics, discrete idealizations of statistical mechanics and dynamic versions of systems such as the Ising model were sometimes close to cellular automata, except for the crucial difference of having randomness built into their underlying rules.) Cellular automata Additives such as rule 90 had implicitly emerged in the studies of the binomial prime modulus coefficient in the nineteenth century, but also appeared in various scenarios, such as the "rickety tree forests" studied around 1970.

However, in the late 1970s, despite all these different directions, research into systems equivalent to cellular automata had largely disappeared. That this should have happened just at the time computers were becoming widely available for exploratory work is ironic.

2.3.2. Definition

The Elementary Cellular Automata are one-dimensional automata where there are two possible states (labeled 0 and 1) and the rule to determine the state of a cell in the next generation depends only on the current state of the cell and its two immediate neighbors. This is one of the simplest possible models of computing. There are $2^3 = 8$ possible configurations for a cell and its two immediate neighbors. The rule that defines the cellular automaton has to specify the resulting state for each of these possibilities, that is, that there are $2^{2^3} = 256$ possible elementary cellular automata. Stephen Wolfram proposed a scheme, known as the Wolfram code, to assign each rule a number from 0 to 255. Each possible current configuration is written in order, 111, 110, ..., 001, 000, and the resulting state for each of these configurations is written in the same order and interpreted as the binary representation of a whole number.

Wolfram represents the one-dimensional cellular automata with two parameters (k, r), where $k = |\Sigma|$ is the number of states and r is the radius of neighbors, therefore,

the domain of the ECA is defined by the parameters (2,1). There are Σ^n different neighborhoods (where n = 2r + 1) and k^{k^n} different rules of evolution.

The elementary cellular automata, being the most basic because they are not produced by difficult rules to follow, tend to show complex comportamients. During the analysis of the elementary rules proposed by Wolfram, interesting patterns have been found in some rules such as rule 110, in which even universal computation has been found and patterns that give way to the study of super-colliders. This exhaustive study of rule 110 was carried out by various people such as Harold V. Mcintosh, Genaro J. Martinez, among others.

2.3.3. Properties

The aspect that most characterizes the CAs is their ability to achieve a series of properties that arise from the local dynamics itself through the passage of time and not from the beginning, applying to the whole system in general. Therefore it is not easy to analyze the global properties of a CA from its beginning, complex by nature, unless by way of simulation, starting from a state or initial configuration of cells and changing at every moment the states of all of them synchronous way.

The cellular automata elementary are simulated by using a grid, where each cell is represented by a square and being black when the state is 1, and white when the state is 0. Simple rules such as rule 0 255 after a few evolutions they fall into a state in which they are maintained over time, while rules such as the 110 evolve in time do not show patterns in a simple way, but many evolutions are required that we would not be able to obtain but by the fact that we can simulate them in a computer. The fact that between the rules there are some simple and other complex, some equivalent to apply to all evolutions a logical operation (example of this are rule 126 and rule 129, equivalent to apply a NOT to the entire space of evolutions), and even some presenting universal computation.

Its dynamism, its evolutions that can fall into 4 classes according to Wolfram, are those that allow the elementary cellular automata to become a true case study.

2.3.4. Wolfram Classification

Wolfram, being the author of these rules, studied his behavior, writing his book entitled "A new kind of science" in 2002, where he talks about everything he had found about these rules, and in it he places its own classification, result of the analysis in the theory of probability and the De Bruijn diagrams and the medium field theory, which is:

1. Class I: Evolution to a uniform state. After a certain number of generations have elapsed, all the cells of the automaton converge to a single state.

2. Class II: Evolution to isolated cyclical states. During the evolution of the automaton, there are certain patterns of behavior that are repeated systematically over time. These patterns can be clearly distinguished on a "backgroundrepresented by a single state which is opposite to that of the cells that represent the pattern of cyclic behavior.

3. Class III: Evolution to broad cyclical states. As in a class II cellular automaton, in a class III cellular automaton there are repetitive patterns of behavior, although not so easily identifiable to the naked eye because the behavior of this can be extremely chaotic, which makes the analysis of this type of automaton.

4. Class IV: Evolution to isolated complex states. This kind of automaton is a combination of classes I, II and III. As in class II automata, there are isolated cyclical behaviors, which makes a uniform "background" distinguishable, which allows to identify such behaviors which are similar to those presented in a class III automaton.

Thus, we can observe examples of this classification in the elementary cellular automata as follows:

2.4. Tools used in the analysis of the ECA

Over the years, from the beginning of the study of cellular automata, when they did not have a name or it was not known that they were actually working with

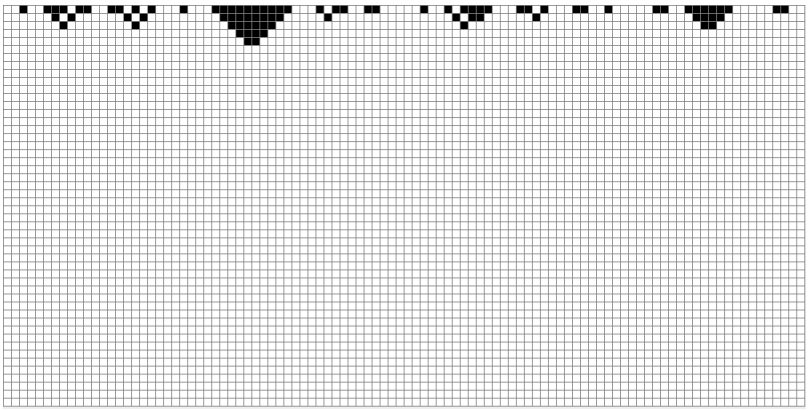


Figura 2.4: Example of an ECA class I: rule 160.

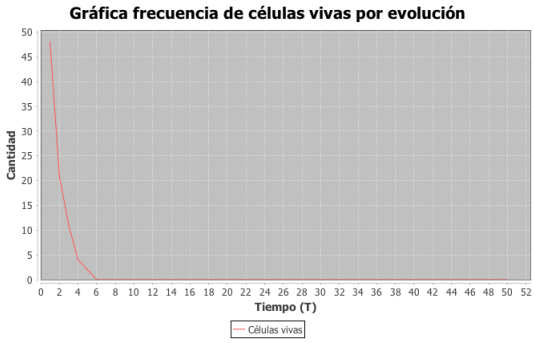


Figura 2.5: Graph of the frequency of living cells by evolution: rule 160

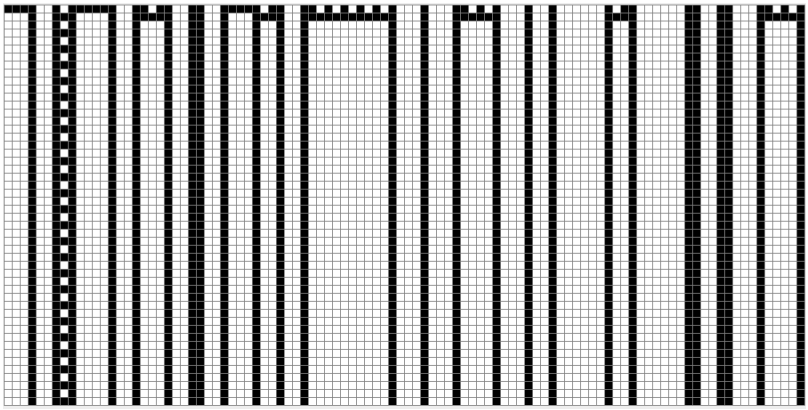


Figura 2.6: Example of an ECA class II: rule 108

one of them, due to their complex behavior, various questions arose about how they could formalize their study. At a certain moment it was decided that it would be

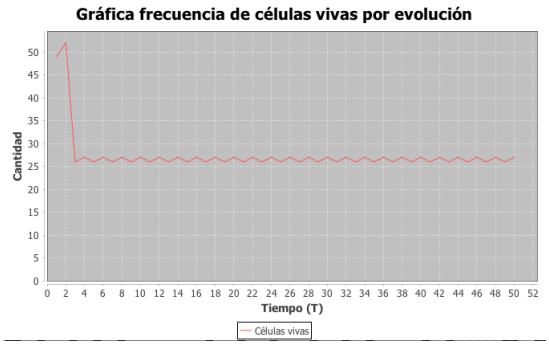


Figura 2.7: Graph of the frequency of living cells by evolution: rule 108

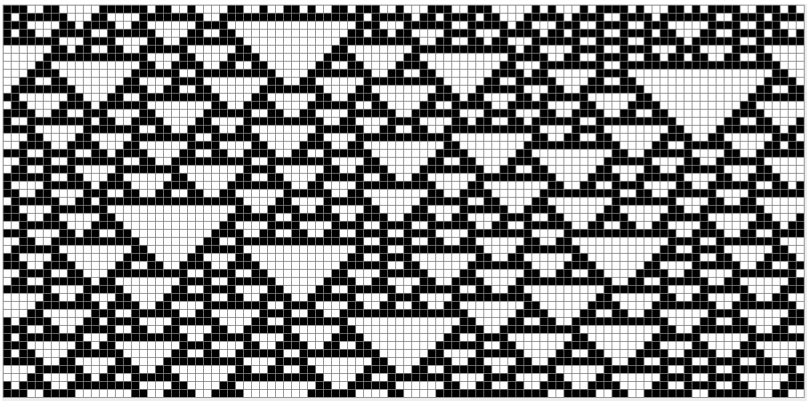


Figura 2.8: Example of an ECA class III: rule 126

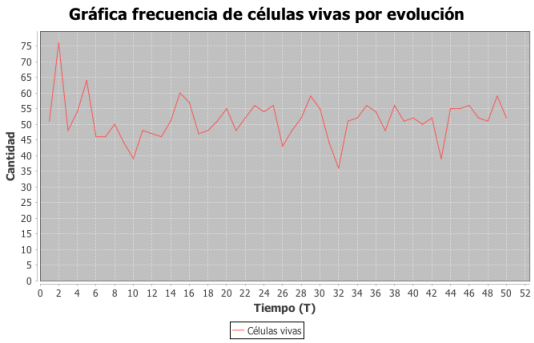


Figura 2.9: Graph of the frequency of living cells by evolution: rule 126

best to analyze them with the current tools for complex systems, giving way to the projection towards tools such as the bubbling diagrams, the Wuensche classification for the attractors and even their De Bruijn diagrams were obtained in various levels

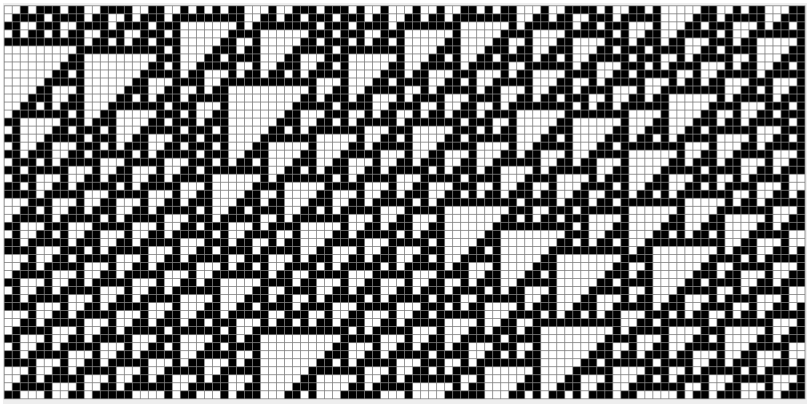


Figura 2.10: Example of an ECA class IV: rule 110

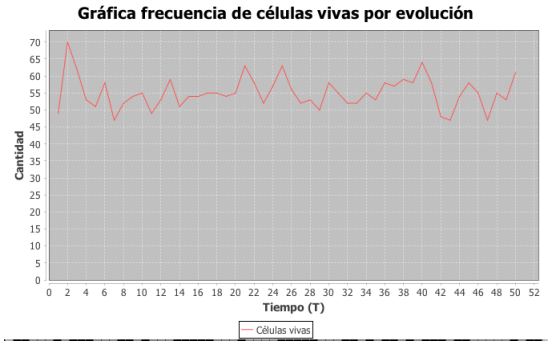


Figura 2.11: Graph of the frequency of living cells by evolution: rule 110

depending on the automaton.

2.4.1. Lyapunov Exponent for ECA

Take rule 126 as an example, let's use an initial configuration of 30 cells in length: 111111100000101010001011001110 and evolve 10 times, obtaining the figure 2.12

Now let's change the initial condition a bit, placing a living cell in cell 15, with the initial configuration 111111100000111010001011001110, let's look at the behavior now in figure 2.13

We can see that the evolutions are different but we do not know exactly how different they are, so let's observe how the evolution of changing a cell changed, as shown in figure 2.14

The part that is highlighted is the cells that are no longer the same as the first

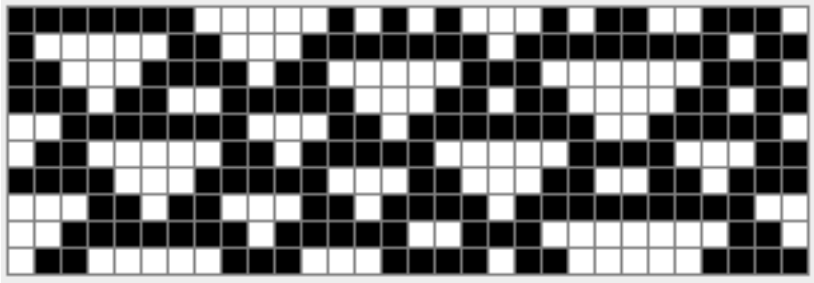


Figura 2.12: Evolution of rule 126

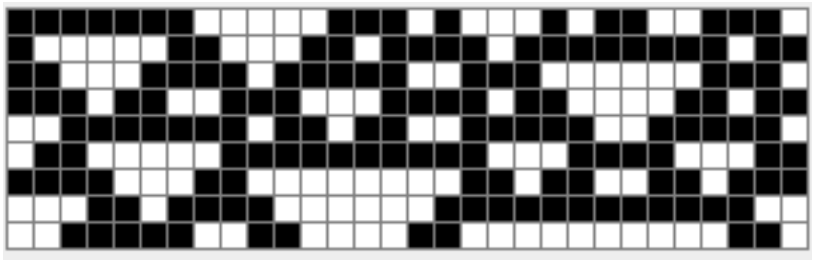


Figura 2.13: Evolution of rule 126 with a change in cell 15

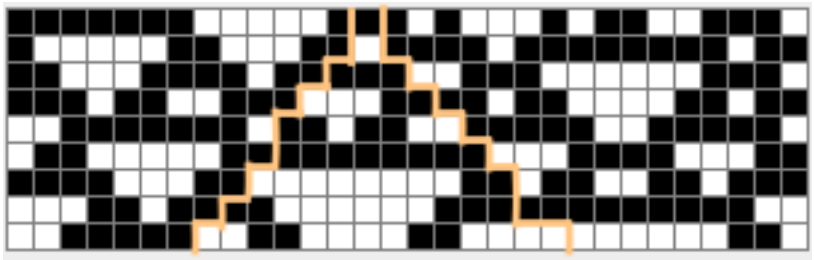


Figura 2.14: Evolution of rule 126 with differences

evolution, so we can see how the chaos expands with a simple change, almost in each evolution there was a change.

2.4.2. Wuensche Classification

Wuensche classifies the attractors of the automatons in 4 classes just like Wolfram, but he is based mainly on the shape of the attractors, his garden of Eden and the size of his cycle. So, we have that the classification is:

1. Class I: They have very short transients, mainly they are attractors of a point

(although possibly they are also periodic attractors) very high in degree and very high density in the leaves (very ordered dynamics). This class has either one or a few states that will be cycling in time, while its Garden of Eden tends to have many states that converge at one point.

2. Class II: They have very short transients, mainly small periodic attractors (but can also have single-point attractors), high in degree and very high density in their leaves. That is, class II have short or single-point courses, but they are distinguished by having lower grade than class I.

3. Class III: They have very long transients, very long periodic attractors, they are low in degree and have low density in their leaves (chaotic dynamics). This type has its ring where they have their states that repeat very large, above the 1000 states, and have low density in their leaves and are low in degree (it does not take long to reach the attractor states).

4. Class IV: They have moderate transients, moderate length of their periodic attractors, are moderate in degree and very moderate density in their leaves (possibly complex dynamics). This class does not have such large rings of states that are repeated over time, nor is their degree high and they are very moderate with respect to their leaves (there are not as many states to reach the attractor states).

2.4.3. The De bruijn Diagrams

The De Bruijn Diagrams are very suitable to describe the rules of evolution of the cellular automata of a dimension, although they were originally used in the theory of the register of changes (the treatment of sequences where their elements overlap each other) . The paths in a de Bruijn diagram can represent chains, configurations, or classes of configurations in the evolutions space.

For a cetral automaton of a dimension of order (k, r), the De Bruijn diagram is defined as a direct graph with k^{2r} vertices and k^{2r+1} edges. The vertices are labeled with the elements of the alphabet of length 2r. An edge directly from vertex i to vertex j, if and only if, the 2r - 1 final symbols of i are the same as the initial 2r - 1 in j forming a vency of 2r + 1 states represented by ij.

The De Bruijn diagram associated with rule 126 is represented in the figure 2.15

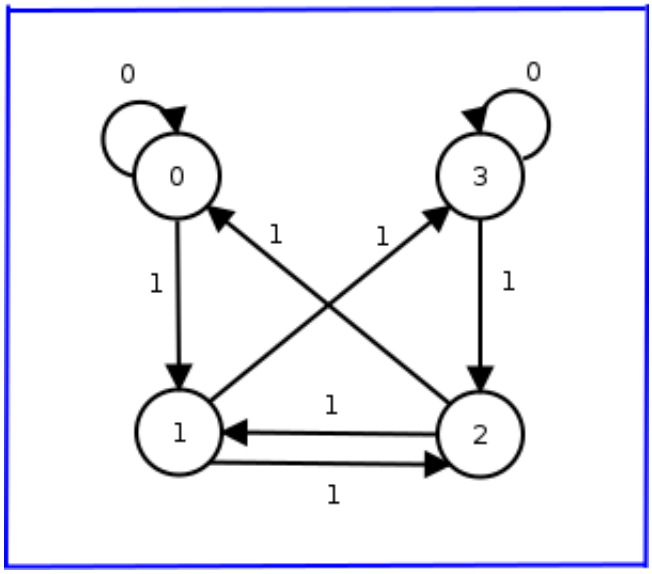


Figura 2.15: The De Bruijn diagram corresponding to rule 126

The following figure shows that there are two neighborhoods evolving towards 0 and six neighborhoods towards 1, so the highest frequency is from state 1, indicating the possibility of having an injective automaton, that is, the existence of configurations of the Garden of Eden. The classical analysis in graph theory has been applied to de Bruijn diagrams to study issues such as reversibility; in another sense, the cycles in the diagram indicate periodic constructions in the evolution of the automaton if the label of the circle is in agreement with the sequence defined by its nodes, taking limited periodic conditions. Take the equivalent construction of a de Bruijn diagram to describe the two-step evolution of rule 126 (now having nodes composed of sequences of 4 symbols).

The k-ways

Suppose there is a path from state i to state j that does not pass through a state greater than k. There are two possible cases to consider:

1. The road does not pass through state k. In this case, the label on the road is in the language of R_{ij}^{k-1}.

2. The road passes through the state k at least once. We can divide the road into several sections. The first one goes from state i to state k without passing through k, the last section goes from state k to k without passing through k, and the remaining intermediate sections go from k to k, without passing through k. Note that if the path traversed the state k only once, then there would be no ïntermediate"stretch, only a path from i to k and a path from k to j. The set of labels for all roads of this type is represented by the regular expression $R_{ik}^{k-1}(R_{kk}^{k-1})^* R_{kj}^{k-1}$. That is, the first expression represents the part of the path that reaches the state k for the first time, the second represents the part that goes from k to zero, one or more at a time, and the third expression represents the part of the path that leaves k for the last time and goes to state j.

If we combine the expressions for the paths of the two previous types, we have the expression:

$R_{ij}^{k} = R_{ij}^{k-1} + R_{ik}^{k-1}(R_{kk}^{k-1})^* R_{kj}^{k-1}$

for the labels of all roads from the state to the state that do not pass through any state greater than k. If we construct these expressions in increasing order of the super indexes, given that each R_{ij}^{k} it only depends on the expressions with the smallest superscript, then all the expressions will be available when we need them.

Then we have R_{ij}^{n} for everything i and j. We can assume that state 1 is the initial state, although acceptance states could be any set of states. The regular expression for the language of the automaton is then the sum (union) of all the expressions R_{ij}^{n} such that state j is a state of acceptance.

2.5. Chaotic Systems

2.5.1. Definition of a chaotic system

The theory of chaos was introduced in ecology by May (974, 1976) and Oster (1976) in the context of real functions of real variable is being studied intensively in recent years and appears in almost all non-linear discrete models.

The first thing that catches our attention is the fact that we live immersed in chaos. In the usual way, we call chaos everything that we are not able to systematize.

The first investigator of chaos was a meteorologist named Edward Lorentz. In 1960 he used a mathematical model to predict time, which consisted of a system of 12 nonlinear equations. The simulation was carried out with a computer, which gave as response a probable behavior of the atmosphere. On one occasion, he wanted to repeat the previous calculations again, for this he re-entered the numbers on the computer, but to save paper and time, he only used 3 decimal numbers instead of 6. The surprising thing was that the result found was totally different to those obtained in the previous time. From the analysis of this situation arose a new theory that is known by the name of chaos theory.

What was really interesting was that very small differences in the initial conditions had a great influence on the final resolution of the problem. To this effect they have the small initial differences after it was given the name of butterfly effect: The movement of a simple butterfly wing today produces a tiny change in the state of the atmosphere. After a certain period of time, the behavior of the atmosphere diverges from what it should have had. So, in a period of one month, a tornado that would have devastated the Indonesian coast is not formed.

Before the appearance of this new theory, there were only two types of known behaviors for a dynamic system: a fixed state, where variables never change, and periodic behavior, where the system is in a "closed loop" and Repeat infinitely.

2.5.2. Properties

Some of the characteristic features of chaotic systems are:

1. They are very sensitive to the initial conditions. A very small change in the initial data results in totally different results.

2. They look like a mess, or random facts, but they are not, there are rules that determine their behavior. Random systems are not chaotic.

2.5.3. Identification of a Chaotic System

Chaos is no more than a disorder only in appearance, it has very little to do with chance. Although they seem to evolve randomly and erratically, these systems actually have an underlying internal order. Therefore, even when they are unpredictable, they are also deterministic. Which means that its future state is determined by its current state and obeys strict natural laws of dynamic evolution. But these systems are so irregular that they never repeat their past behavior, even in an approximate way.

Chaos seems to be part of the very structure of matter and is closely linked to the phenomena of self-organization, since the system can spontaneously and recurrently jump from one state to another of greater complexity and organization.

2.6. Complex Systems

2.6.1. Definition of a Complex System

The Complex Systems are characterized mainly because their behavior is unpredictable. However, complexity is not synonymous with complication: the latter refers to something tangled, tangled, difficult to understand. In reality, and for the moment, there is no precise and absolutely accepted definition of what a complex system is, but there may be some common peculiarities.

2.6.2. Properties

1. First, it is composed of a large number of relatively identical elements. For example, cells in an organism, or people in a society.

2. Secondly, the interaction between its elements is local and originates an emergent behavior that can not be explained from these elements taken in isolation. A desert may contain billions of grains of sand, but its interactions are exceedingly simple compared to those that occur in bees in a swarm.

3. Finally, it is very difficult to predict its future dynamic evolution; that is, it is practically impossible to predict what will happen beyond a certain time

horizon.

In nature you can find a lot of examples of complex systems ranging from physics to neurology, from economics to molecular biology, from sociology to mathematics. For this reason, this kind of systems is not a rare or exceptional case but is manifested in the vast majority of the phenomena that are observed daily. However, and despite its great diversity and abundance, generic dynamic behaviors can be identified, regardless of their nature (physical, chemical, biological or social); among them, the laws of growth, self-organization and emerging collective processes.

2.6.3. Identification of a Complex System

Most complex systems are unstable, they remain delicately balanced. Any minimal variation between its component elements can change, in an unpredictable way, the interrelationships and, therefore, the behavior of the entire system. Thus, the evolution of this class of systems is characterized by fluctuation, a situation in which order and disorder alternate constantly. Their evolutionary states do not pass through continuous and gradual processes, but they happen through reorganizations and jumps. Each new state is just a transition, a period of "entropic rest"

Capítulo 3

Rule 126 of ACE

3.1. Background

The cellular automata, as mentioned in the previous chapter, arose long ago, but in the case of elementals the first person who spoke of them was Stephen Wolfram in 1983, and since that date several studies have been made about the same. Rule 126 has a chaotic behavior, so Wolfram classified it as class III, although adding an extension (memory), it can be seen that the rule behaves like a class IV. It was demonstrated in the same way that the rule generates a regular language in the book " Scaling Phenomena in Disordered Systems " written in 1991 by Roger Pynn and Arne Skjeltorp, where it shows the attractors of the rule and how they generate a regular language. From there the studies of the rule have been diverse, from the analysis of its de Bruijn diagrams to its extension with memory, arriving at a hypothesis that this rule has characteristics similar to the rule 110. Harold V. McIntosh made an exhaustive exploration of rule 110, leaving gliders and gliders-guns found in said rule, while Genaro J. Martínez decided to explore rule 126 with memory of 4 generations, writing articles about it, but leaving the exploration of the elementary rule unfinished, it is for that is what is required to do the complete exploration of the rule first, using the most common tools, as well as to describe why it actually generates a regular language.

3.2. Definition of rule 126

Rule 126 is one of the elementary rules of cellular automata introduced by Stephen Wolfram in 1983 (Wolfram 1983, 2002). Specifies the next color in a cell, depending on its color and its immediate neighbors. The results of this rule are encoded in the binary representation $126 = 01111110_2$.

The definition of rule 126 graphically is that of the figure 3.1

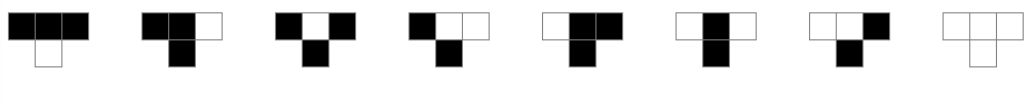


Figura 3.1: The definition of rule 126 according to its left and right neighbors.

We can generalize this rule in the following way: when the cell that is being evaluated, as well as its left and right neighbors have the same state(either 0 or 1), then the next evolution will be a dead cell(0), otherwise it will be a living cell(1). In this way we avoid using the 8 conditions that define it, although we can also use its totalistic function that is not more than the sum of the current values of the 3 cells that are used to be evaluated. Imagine that our combination to evaluate is 011, then its totalistic function will be $0 + 1 + 1 = 2$, since the values are added, so we can say that if the totalistic function of the combination we are evaluation is 0 or 3, then the next evolucion will be 0, otherwise it will be 1.

Evolving the rule sometimes in time with a central living cell we can observe what is shown in the figure 3.2

As we can see, rule 126 under these initial conditions does not seem to show random and complex behaviors. Now, adding a random initial condition and again evolving we can observe the evolution of the figure 3.3

As we can see, the rule under random conditions no longer shows the same uniform behavior as before.

3.3. Propiedades

Given a sequence of 1 and 0, such as that generated by an automaton that evolves from random initial conditions, a discrete Fourier transfromada can be performed

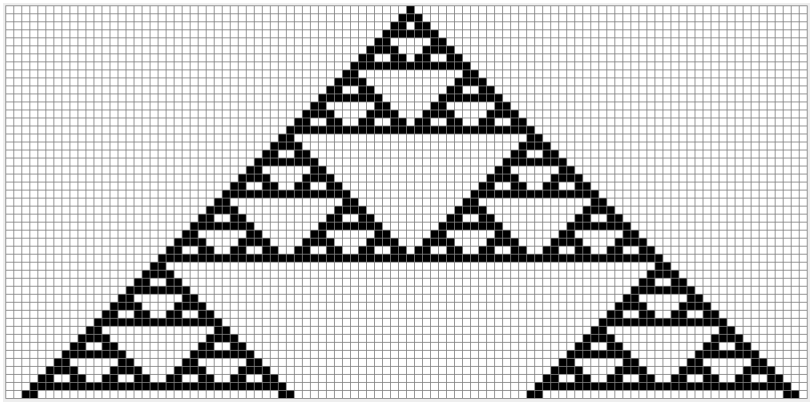


Figura 3.2: Evolution of the rule 126 up to t = 50 with a living cell.

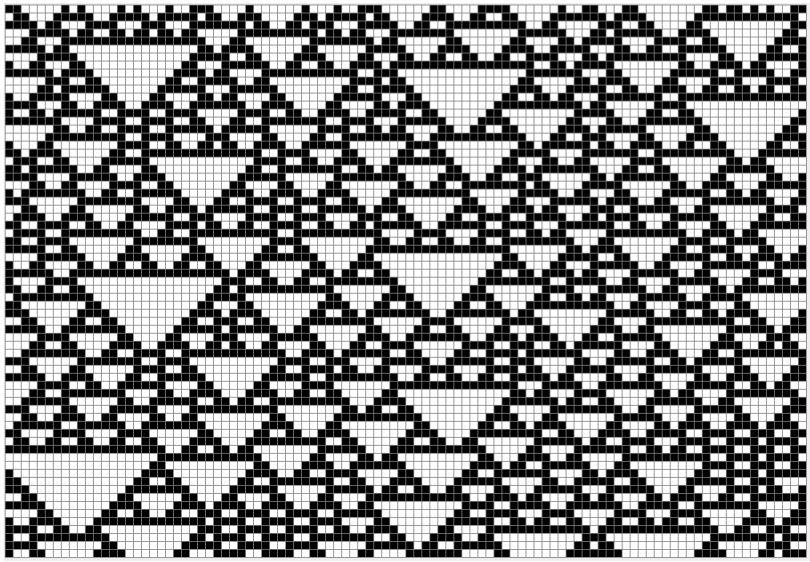


Figura 3.3: Evolution of the rule 126 up to t = 70 with 30 % of its living cells.

to determine which combination of sinuisoidal functions best approximates that sequence. Particularly when the automaton has non-trivial attractors, this spectrum may exhibit a variety of characteristics, such as those seen in rule 126. This power spectrum is no more than the calculation of the discrete Fourier transform at various times t.

The density of the evolutions is shown in the figure 3.4 with gray tones:

We can ask ourselves how the density of the black cells changes as the automaton evolves. In this image, each shade of gray represents a particular density in a parti-

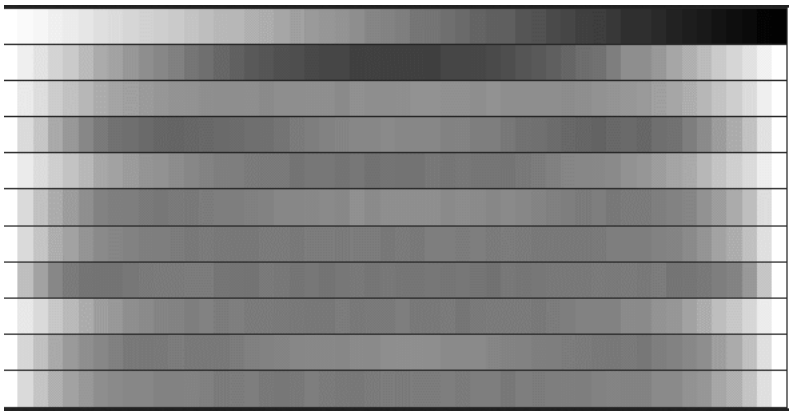


Figura 3.4: Density of the changes in rule 126.

cular line of cells. The upper line represents a set of initial conditions with increasing density, and each step below shows the density of the state of the automaton after evolving from those initial conditions. The number of different initial conditions used is finite, but large enough to appear relatively continuous on the scales used here.

3.4. Classification of rule 126

According to Wolfram's classification, rule 126 enters class III. We have already commented that class III denotes chaotic behavior, and as in this case the rule that is being studied falls into this classification it is time to see why. First, we will take a random evolution of cells as seen in the figure 3.5

As we can see in the figure with random combination, the patterns found are almost unpredictable, some are repeated but others definitely only happen once, this behavior is known as chaotic, due to the complexity of describing it and finding a function for its evolution.

3.5. The De Bruijn Diagram for rule 126

For rule 126, let's get the k-ways of your De Bruijn diagram. Since this diagram only has 4 states, we will take up k = 3 as our maximum, and we will start with k = 0. So the k-ways are the following:

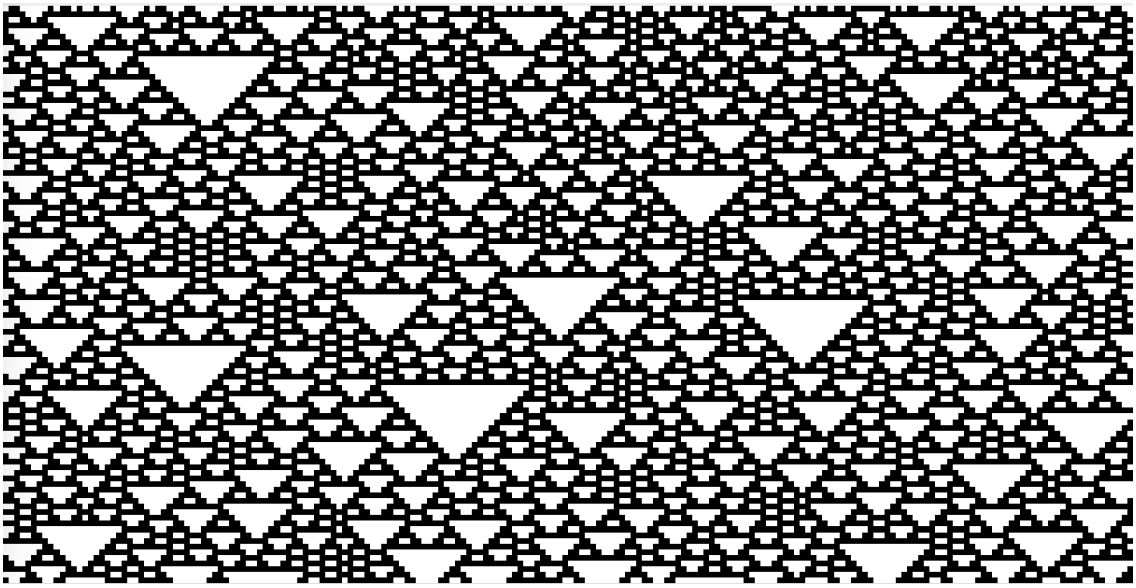


Figura 3.5: Evolution of the random 126 rule.

For k = 0

$R^0_{00} = 0$

$R^0_{01} = 1$

$R^0_{02} = \emptyset$

$R^0_{03} = \emptyset$

$R^0_{10} = \emptyset$

$R^0_{11} = \emptyset + \epsilon = \epsilon$

$R^0_{12} = 1$

$R^0_{13} = 1$

$R^0_{20} = 1$

$R^0_{21} = 1$

$R^0_{22} = \emptyset + \epsilon = \epsilon$

$R^0_{23} = \emptyset$

$R^0_{30} = \emptyset$

$R^0_{31} = \emptyset$

$R^0_{32} = 1$

$R^0_{33} = 0$

For k = 1

$R^1_{00} = R^0_{00} + R^0_{01}(R^0_{11})^* R^0_{10} = 0 + 1(\epsilon)^*\emptyset = 0 + \emptyset = 0$
$R^1_{01} = R^0_{01} + R^0_{01}(R^0_{11})^* R^0_{11} = 1 + 1(\epsilon)^*\epsilon = 1 + 1 = 1$
$R^1_{02} = R^0_{02} + R^0_{01}(R^0_{11})^* R^0_{12} = \emptyset + 1(\epsilon)^*1 = 11$
$R^1_{03} = R^0_{03} + R^0_{01}(R^0_{11})^* R^0_{13} = \emptyset + 1(\epsilon)^*1 = 11$
$R^1_{10} = R^0_{10} + R^0_{11}(R^0_{11})^* R^0_{10} = \emptyset + \epsilon(\epsilon)^*\emptyset = \emptyset$
$R^1_{11} = R^0_{11} + R^0_{11}(R^0_{11})^* R^0_{11} = \epsilon + \epsilon(\epsilon)^*\epsilon = \epsilon$
$R^1_{12} = R^0_{12} + R^0_{11}(R^0_{11})^* R^0_{12} = 1 + \epsilon(\epsilon)^*1 = 1 + 1 = 1$
$R^1_{13} = R^0_{13} + R^0_{11}(R^0_{11})^* R^0_{13} = 1 + \epsilon(\epsilon)^*1 = 1$
$R^1_{20} = R^0_{20} + R^0_{21}(R^0_{11})^* R^0_{10} = 1 + 1(\epsilon)^*\emptyset = 1$
$R^1_{21} = R^0_{21} + R^0_{21}(R^0_{11})^* R^0_{11} = 1 + 1(\epsilon)^*\epsilon = 1$
$R^1_{22} = R^0_{22} + R^0_{21}(R^0_{11})^* R^0_{12} = \epsilon + 1(\epsilon)^*1 = \epsilon + 11$
$R^1_{23} = R^0_{23} + R^0_{21}(R^0_{11})^* R^0_{13} = \emptyset + 1(\epsilon)^*1 = 11$
$R^1_{30} = R^0_{30} + R^0_{31}(R^0_{11})^* R^0_{10} = \emptyset + \emptyset(\epsilon)^*\emptyset = \emptyset$
$R^1_{31} = R^0_{31} + R^0_{31}(R^0_{11})^* R^0_{11} = \emptyset + \emptyset(\epsilon)^*\epsilon = \emptyset$
$R^1_{32} = R^0_{32} + R^0_{31}(R^0_{11})^* R^0_{12} = 1 + \emptyset(\epsilon)^*1 = 1$
$R^1_{33} = R^0_{33} + R^0_{31}(R^0_{11})^* R^0_{13} = 0 + \emptyset(\epsilon)^*1 = 0$

For k = 2

$R^2_{00} = R^1_{00} + R^1_{02}(R^1_{22})^* R^1_{20} = 0 + 11(\epsilon + 11)^*1 = 0 + (11)^*1$
$R^2_{01} = R^1_{01} + R^1_{02}(R^1_{22})^* R^1_{21} = 1 + 11(\epsilon + 11)^*1 = 1 + (11)^*1$
$R^2_{02} = R^1_{02} + R^1_{02}(R^1_{22})^* R^1_{22} = 11 + 11(\epsilon + 11)^*(\epsilon + 11) = (11)^*$
$R^2_{03} = R^1_{03} + R^1_{02}(R^1_{22})^* R^1_{23} = 11 + 11(\epsilon + 11)^*11 = (11)^*$
$R^2_{10} = R^1_{10} + R^1_{12}(R^1_{22})^* R^1_{20} = \emptyset + 1(\epsilon + 11)^*1 = 1(11)^*1$
$R^2_{11} = R^1_{11} + R^1_{12}(R^1_{22})^* R^1_{21} = \epsilon + 1(\epsilon + 11)^*1 + 1(11)^*1$
$R^2_{12} = R^1_{12} + R^1_{12}(R^1_{22})^* R^1_{22} = 1 + 1(\epsilon + 11)^*(\epsilon + 11) = 1 + 1(11)^*$
$R^2_{13} = R^1_{13} + R^1_{12}(R^1_{22})^* R^1_{23} = 1 + 1(\epsilon + 11)^*11 = 1 + 1(11)^*$
$R^2_{20} = R^1_{20} + R^1_{22}(R^1_{22})^* R^1_{20} = 1 + (\epsilon + 11)(\epsilon + 11)^*1 = 1 + (11)^*1$
$R^2_{21} = R^1_{21} + R^1_{22}(R^1_{22})^* R^1_{21} = 1 + (\epsilon + 11)(\epsilon + 11)^*1 = 1 + (11)^*1$
$R^2_{22} = R^1_{22} + R^1_{22}(R^1_{22})^* R^1_{22} = (R^1_{22})^* = (11)^*$
$R^2_{23} = R^1_{23} + R^1_{22}(R^1_{22})^* R^1_{23} = 11 + (\epsilon + 11)(\epsilon + 11)^*11 = (11)^*$
$R^2_{30} = R^1_{30} + R^1_{32}(R^1_{22})^* R^1_{20} = \emptyset + 1(\epsilon + 11)^*1 = 1(11)^*1$
$R^2_{31} = R^1_{31} + R^1_{32}(R^1_{22})^* R^1_{21} = \emptyset + 1(\epsilon + 11)^*1 = 1(11)^*1$
$R^2_{32} = R^1_{32} + R^1_{32}(R^1_{22})^* R^1_{22} = 1 + 1(\epsilon + 11)^*(\epsilon + 11) = 1 + 1(11)^*$

$R^2_{33} = R^1_{33} + R^1_{32}(R^1_{22})^* R^1_{23} = 0 + 1(\epsilon + 11)^* 11 = 0 + 1(11)^*$

For k = 3

$R^3_{00} = R^2_{00} + R^2_{03}(R^2_{33})^* R^2_{30} = 0 + (11)^*1 + (11)^*(0 + 1(11)^*)^*1(11)^*1$

$R^3_{01} = R^2_{01} + R^2_{03}(R^2_{33})^* R^2_{31} = 1 + (11)^*1 + (11)^*(0 + 1(11)^*)^*1(11)^*1$

$R^3_{02} = R^2_{02} + R^2_{03}(R^2_{33})^* R^2_{32} = (11)^* + (11)^*(0 + (11)^*)^*(1 + (11)^*)^*$

$R^3_{03} = R^2_{03} + R^2_{03}(R^2_{33})^* R^2_{33} = (11)^* + (11)^*(0 + (11)^*)^*(0 + (11)^*)^*$

$R^3_{10} = R^2_{10} + R^2_{13}(R^2_{33})^* R^2_{30} = 1(11)^*1 + (1 + 1(11)^*)(0 + 1(11)^*)^*1(11)^*1$

$R^3_{11} = R^2_{11} + R^2_{13}(R^2_{33})^* R^2_{31} = 1(11)^*1 + (1 + 11(11)^*)(0 + 1(11)^*)^*1(11)^*1$

$R^3_{12} = R^2_{12} + R^2_{13}(R^2_{33})^* R^2_{32} = 1 + 1(11)^* + (1 + 1(11)^*)(0 + 1(11)^*)^*(1 + 1(11)^*)$

$R^3_{13} = R^2_{13} + R^2_{13}(R^2_{33})^* R^2_{33} = 1 + 1(11)^* + (1 + 1(11)^*)(0 + 1(11)^*)^*(0 + 1(11)^*)$

$R^3_{20} = R^2_{20} + R^2_{23}(R^2_{33})^* R^2_{30} = 1 + (11)^*1 + (11)^*(0 + 1(11)^*)^*1(11)^*1$

$R^3_{21} = R^2_{21} + R^2_{23}(R^2_{33})^* R^2_{31} = 1 + (11)^*1 + (11)^*(0 + 1(11)^*)^*1(11)^*1$

$R^3_{22} = R^2_{22} + R^2_{23}(R^2_{33})^* R^2_{32} = (11)^* + (11)^*(0 + 1(11)^*)^*(1 + 1(11)^*)$

$R^3_{23} = R^2_{23} + R^2_{23}(R^2_{33})^* R^2_{33} = (11)^* + (11)^*(0 + 1(11)^*)^*(0 + 1(11)^*)$

$R^3_{30} = R^2_{30} + R^2_{33}(R^2_{33})^* R^2_{30} = 1(11)^*1 + (0 + 1(11)^*)(0 + 1(11)^*)^*1(11)^*1$

$R^3_{31} = R^2_{31} + R^2_{33}(R^2_{33})^* R^2_{31} = 1(11)^*1 + (0 + 1(11)^*)(0 + 1(11)^*)^*1(11)^*1$

$R^3_{32} = R^2_{32} + R^2_{33}(R^2_{33})^* R^2_{32} = 1 + 1(11)^* + (0 + 1(11)^*)(0 + 1(11)^*)^*(1 + 1(11)^*)$

$R^3_{33} = R^2_{33} + R^2_{33}(R^2_{33})^* R^2_{33} = (0 + 1(11)^*)^*$

As we can see in regular expressions, most are not large and some can not even be implemented (those that do not allow generating at least one string of 2 characters in length), also those that generate either a string of only zeros or some do not present an interesting behavior due to the definition of rule 126, however, the last regular expression is the one that allows us to obtain a complex and interesting behavior.

Getting some random strings from the regular expression $R^3_{33} = R^2_{33} + R^2_{33}(R^2_{33})^* R^2_{33} = (0 + 1(11)^*)^*$, the evolutions of the figures 3.6 and 3.7 are obtained.

3.6. Wuensche classification for rule 126

As we saw in the section of the attractors of the rule 126, with short chains we can not observe a chaotic behavior, but when increasing the length and arriving until l = 24 we can see something totally different. In the descriptive of Wuensche's classification we had what happened with class III, that its trees were not so full of

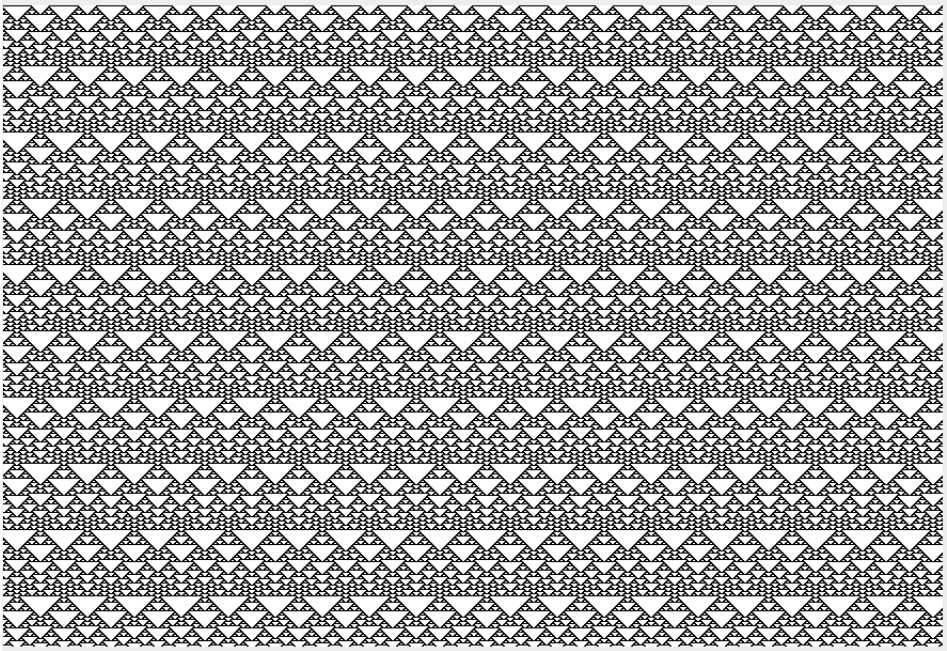


Figura 3.6: Graph of the regular expression.

history or large cycles, but they had a large number of possible routes to reach one of the states attractor.

As observed for chains of length equal to 15, the largest attractor has a great history, however, when it reaches the attractor state it does not have a large number of states in which it cycles, only one. In that figure we can see that the attractor then meets Wuensche's classification. We can take another example within the attractors: when the chain is of length equal to 24 we see that the cycles are increasing but not significantly, a class IV has cycles of at least 100 states, in our case our attractor states when they are a cycle oscillate between 10 - 20 states, but the density they have in the leaves is high, such is the case of these attractors, which have many states that converge to one, and this is one of the ways to reach the states that make up the attractor, so we can conclude that the attractors of the rule 126 meet the classification of Wuensche and this is clearly observed when we evolved with large chains, from l = 13 you can see perfectly its chaotic behavior.

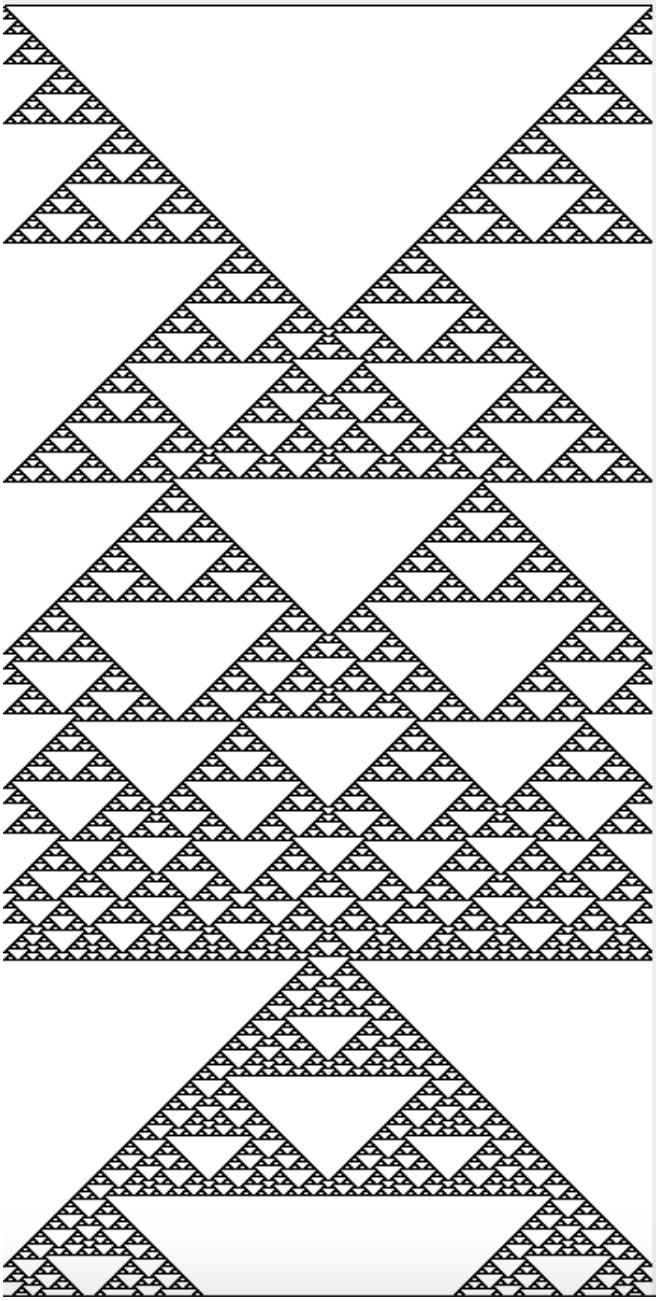


Figura 3.7: Graph of the regular expression.

Capítulo 4

Rule 126 with memory of 4 generations

4.1. Background

While it is true that automata have characteristics that can turn them into complex systems, they only care about the present evocation to generate the next, completely forgetting what they had generated in the past, which is why we call them ahistorical. The case of the automatons with memory is that they take certain evolutions past in time and by means of a function they calculate a temporary ring, said ring is later evaluated with the rule of the automaton and the obtained result is added to the following evolution, thus allowing Your story is important.

As with any automaton, rule 126 can be added the function with memory that is desired to observe its behavior, in this work the rule of majority will be used taking 4 generations, this function was analyzed and shown to the public by Ramón Alonso -Sanz in his book "Cellular Automata with Memory" in 2009, where he shows the automaton and its evolution and compares it with other elementary cellular automata with applied memory. When this automaton evolves to more than 500 evolutions, the automaton shows structures that seem to move and repeat themselves in two dimensions and are surrounded by a repetitive structure called the periodic background, said structures were called gliders and the way in which they can be observed can be observed. they move through the automaton and what happens if

they collide with some other structure than their periodic background. After work by Alonso-Sanz, in 2011 together with Genaro J. Martínez and Andrew Adamatzy they wrote an article entitled "Rule 126 Case Studying", where they describe appreciable gliders and some simulated collisions.

The reason why the memory was taken up to 4 generations, is that when $\tau = t-2$, generates the same evolution that if memory is not used, and in the case of $\tau = t-3$, these patterns are not observed during evolution, so increasing memory in the same way shows patterns as gliders but it is not the object of study of this thesis.

4.2. Rule 126 with memory: majority function

Rule 126 with memory of 4 generations uses the majority function, which defines that having a greater number of zeros or ones taking 4 generations back, then the respective value will be used in the temporary ring, in case of being 2 zeros and 2 ones, the value that is in $\tau = t - 1$. When applied to rule 126, we can observe the evolution of the figure 4.1 for a space of 500 cells per 500 generations.

We can see inside the graph that some lines stand out more than others, and it seems that they are surrounded by states that allow them to stand out, these lines that move from left to right, vice versa or from top to bottom are the gliders and the states that we they allow you to observe them and that you are around them is known as the periodic background. The first 4 evolutions of the attractor are applied the 126 of the ECA directly, from the 5th is when the memory function and the temporary ring is applied. In the figure 4.2 we can see an evolution of the rule with memory using a random condition with a density of 75 % of dead cells and 25 % of living cells for a space of 500 cells for 500 generations.

4.3. Properties

One of the properties of the 126 rule with memory of 4 generations is that it has several gliders with a period equal to 2 or 5, that is, every 2 or 5 evolutions in time the pattern will be repeated but a few cells moved to the left or right. These patterns can be rotated in the same way, so we can start with any of the 5 evolutions that are

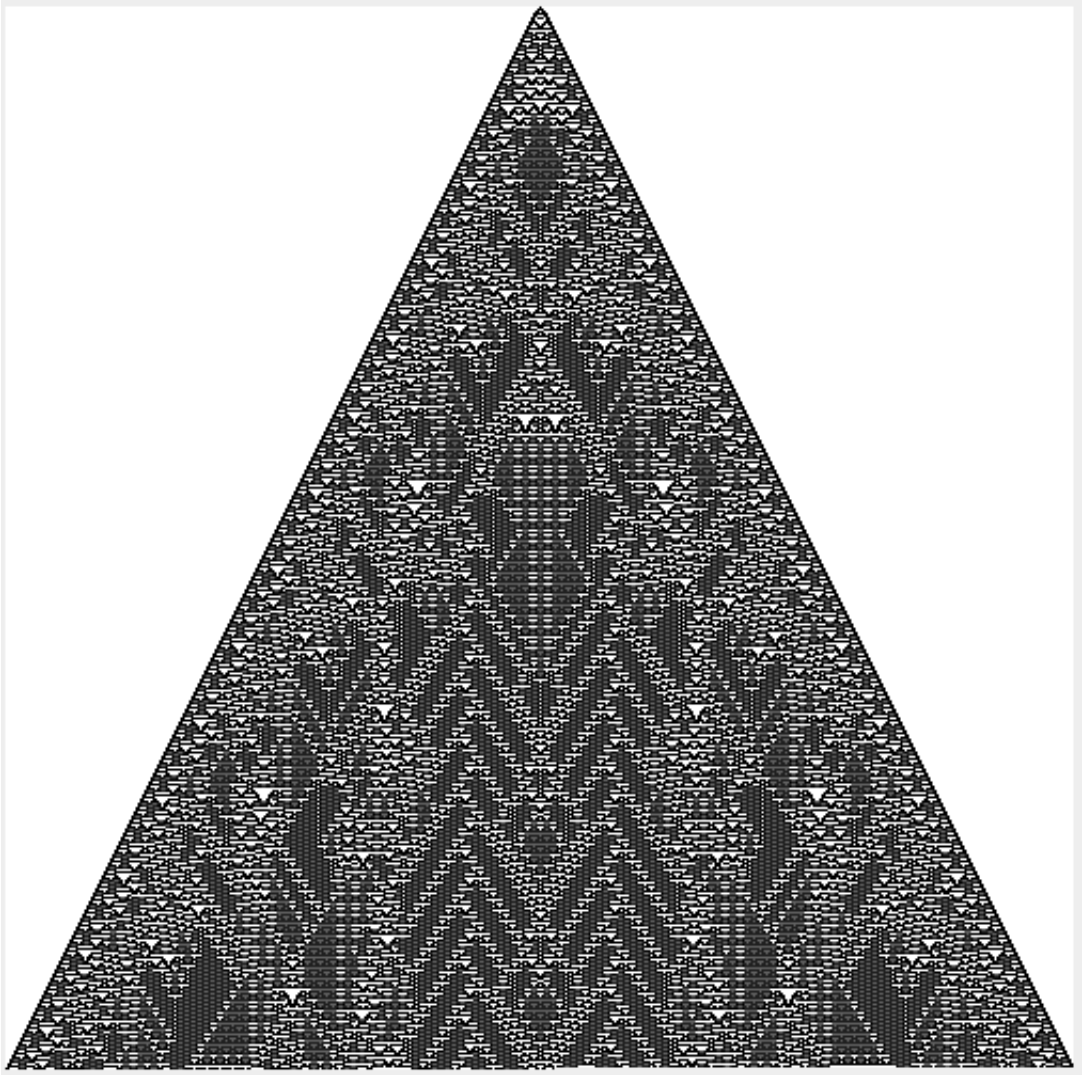


Figura 4.1: Rule 126 with memory of 4 generations with a central living cell

repeated over time. To have control over these patterns, you must specify at least 4 evolutions that generate these patterns, so that in the 5th evolution you begin to use the memory and the glider is not destroyed.

As the 126 rule with memory is an extension of the elementary, all the periodic funds that are found, the gliders and filters, also apply for the rule without memory. This allows us to observe patterns that if we tried to see them only using the rule without memory we would never see them or it would be very difficult, that is why

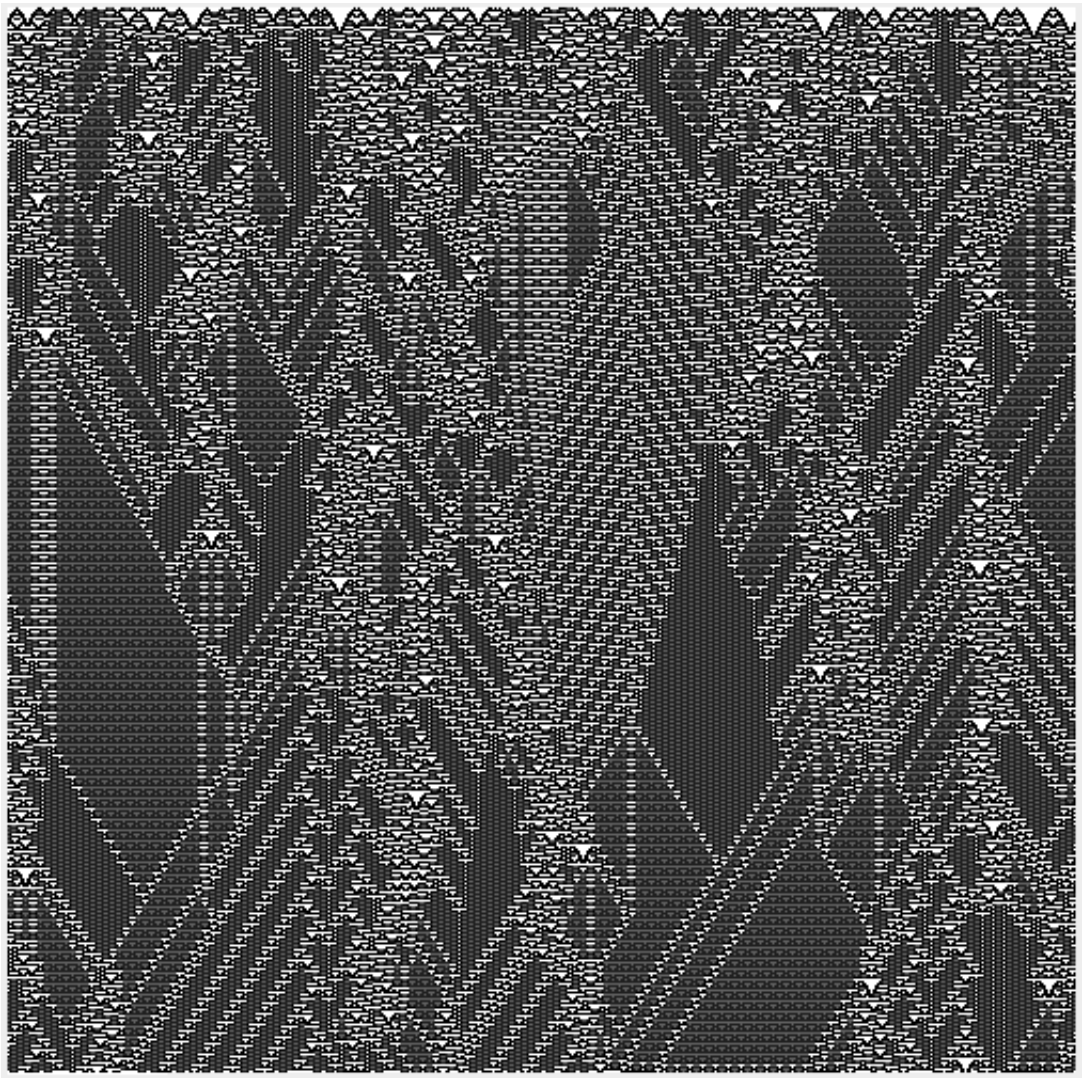


Figura 4.2: Rule 126 with memory of 4 generations with a random condition

it is important to study them with memory.

There is a reverse procedure to pass a function with memory to one without memory and this is to extend the ring until no combination has different states in its next evolution, this allows us to transfer this function to one without memory for analysis. The 126 rules with memory of 2 generations generates the same pattern as that of no memory, but from 3 cells it is almost impossible to determine the function without memory because even extending the ring to a radius of 35, the ambiguity

remains unaffected and this already generates 2^{35} possible combinations, which can be programmed next to the 2^3 Wolfram combinations really is inefficient to continue treating the automaton with its memory function.

4.4. Elements within the evolution space

Within the evolution space of rule 126 with memory, we find some elements that seem to repeat themselves over time and observe how they move and maintain their structure, as well as the background in which they travel.

Within this type of structure, we find 4 types within the rule 126, which are:

1. **Tile for filtering:** They are a set of cells that look like a mosaic that repeats infinitely, but allow the existence of gliders, gliders-gun and stationary structures, with the tile for filtering the gliders that exist maintain their shape thanks to that they are inside the background, outside his structure would not be maintained. Within rule 126 we find 2 tiles for filtering.

2. **Glider:** This type of structure is characterized by being a set of cells that seem to move either to the left or to the right and maintain their shape. Within rule 126 we find 2 different gliders.

3. **Glider-gun:** They are a set of cells that seem to "shoot" gliders, we can also know them as generators of gliders. Within rule 126 we find a glider-gun.

4. **Still-life:** Still-life are similar to gliders, except that with the difference that they do not move to the left or to the right, they remain in the same position. Within rule 126 we find two still-life

In the following sections, these structures will be described and displayed within the evolutions space of rule 126 with memory, as well as some properties they possess.

4.4.1. Tiles for filtering

The tiles for filtering are "mosaics"that allow the existence of gliders, gliders-gun and stationary structures, allow them to maintain their shape and interact with each

other, hence the importance of knowing the various periodical backgrounds that rule 126 has with memory . Within the evolution space we find two tiles for filtering that allow the existence of gliders.

Tile for filtering 1

The 1st tile for filtering is the shown in figure 4.3

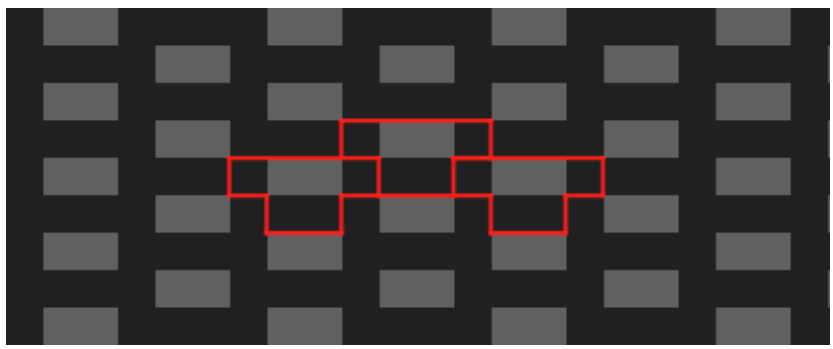


Figura 4.3: Tile for filtering 1(fp_1) in the evolutionary space

As we can see, it looks like a mosaic filled with white rectangles that are separated by black lines and also alternate in order. However, the mosaic that defines our background is a T that is indicated in red.

The regular expression that defines the tile for filtering fp_1 is the table 4.1.

Line	Cells	Phase
1001	4	1
11	2	2

Cuadro 4.1: Lines of cells to form the tile for filtering 1

Tile for filtering 2

The 2nd tile for filtering is the shown in figure 4.4

As we can see, it looks like a mosaic filled with T's separated by white lines, this background is a bit more complex than the periodic background 1, and each one has

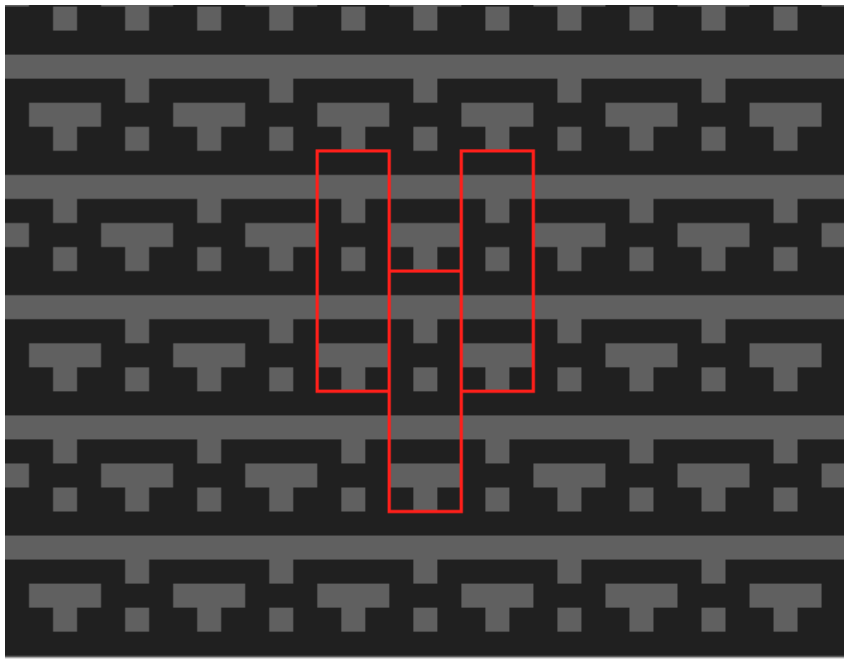


Figura 4.4: Tile for filtering $2(fp_2)$ in the evolutionary space

its set of cells that form it. The mosaic that defines it is a rectangle of 3 cells wide by 10 high, which is shown in red.

The regular expression that defines the tile for filtering fp_2 is the table 4.2.

Line	Cells	Phase
111	3	1
000	3	2
101	3	3
111	3	4
101	3	5
111	3	6
000	3	7
111	3	8
000	3	9
101	3	10

Cuadro 4.2: Cell lines to form the tile for filtering 2

4.4.2. Gliders

The gliders as we saw, are a set of cells that move in the space of evolutions and maintain a form, can move either to the left or to the right and can encompass more than one evolution, in this section we will present the two gliders found in rule 126 with memory. Each of the gliders will be labeled with a "g" followed by the glider number.

Glider g_1

The glider 1 can be seen in the figure 4.5.

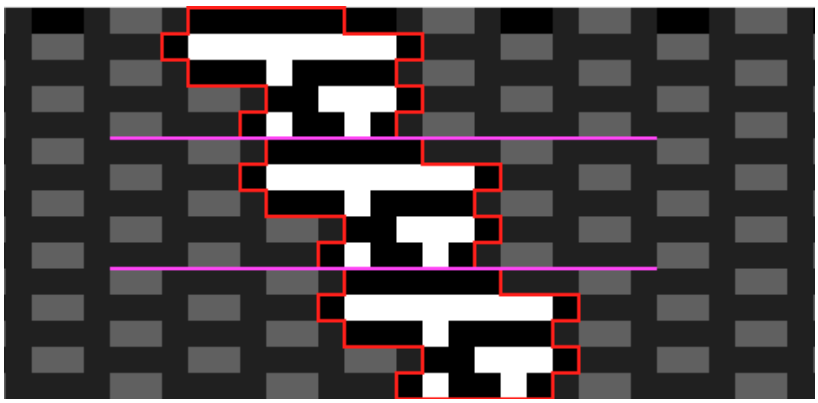


Figura 4.5: Glider 1(g_1) moving within the space of evolutions

As we can see in the figure, glider 1 has 5 evolutions to be formed and also lives within the periodic background 1. Let's see now which cells form the glider in the table 4.3

Line	Cells	Phase
111111	6	1
1000000001	10	2
11101111	8	3
110001	6	4
101101	6	5

Cuadro 4.3: Lines of cells to form the glider 1

Glider g_2

The glider 2 can be seen in the figure 4.6.

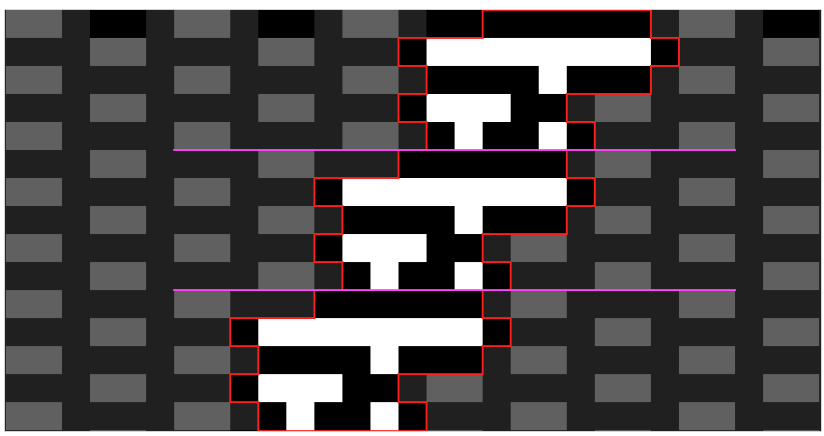


Figura 4.6: Glider 2(g_2) moving within the space of evolutions

As seen in the figure, the glider 2 is similar to glider 1, it seems as if it were its mirror. This glider moves to the left and is also composed of 5 evolutions and lives within the periodic background 1. Let's see its table of properties.

Glider g_3

The glider 3 can be seen in the figure 4.7.

Line	Cells	Phase
111111	6	1
1000000001	10	2
11110111	8	3
100011	6	4
101101	6	5

Cuadro 4.4: Lines of cells to form the glider 2

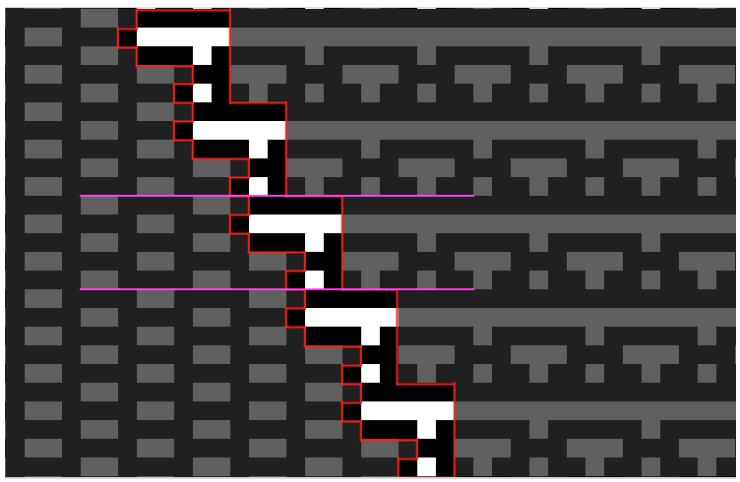


Figura 4.7: Glider 3(g_3) moving within the space of evolutions

As seen in the figure, the glider 3 coexists between the periodic funds 1 and 2, so we can use it as a bridge between these two spaces. The regular expression that defines it is described in the table 4.5

Glider g_4

The glider 4 can be seen in the figure 4.8.

As seen in the figure, glider 4 coexists betweentiles for filtering 1 and 2, so we can use it as a bridge between these two spaces. The regular expression that defines it is described in the table 4.6

Glider g_5

The glider 5 can be seen in the figure 4.9.

Line	Cells	Phase
11111	5	1
100000	6	2
11101	5	3
11	2	4
101	3	5

Cuadro 4.5: Lines of cells to form the glider 3

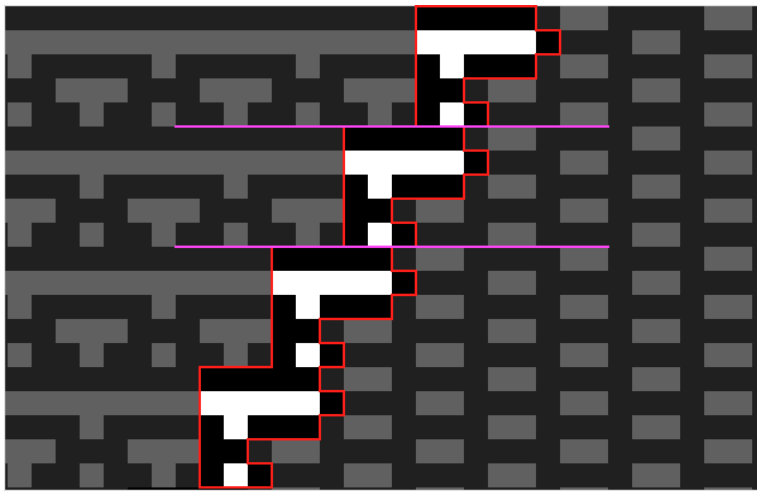


Figura 4.8: Glider 4(g_4) moving within the space of evolutions

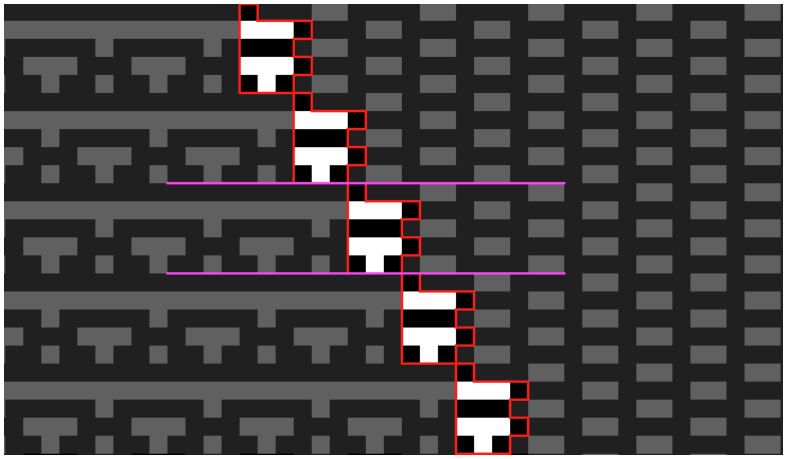


Figura 4.9: Glider 5(g_5) moving within the space of evolutions

Line	Cells	Phase
11111	5	1
000001	6	2
10111	5	3
11	2	4
101	3	5

Cuadro 4.6: Cell lines to form the glider 4

As seen in the figure, the glider 5 coexists between the tiles for filtering 1 and 2, so we can use it as a bridge between these two spaces. The regular expression that defines it is described in the table 4.7

Line	Cells	Phase
1	1	1
0001	4	2
111	3	3
0001	4	4
101	3	5

Cuadro 4.7: Lines of cells to form the glider 5

Glider g_6

The glider 6 can be seen in the figure 4.10.

As seen in the figure, glider 6 coexists between periodic funds 1 and 2, so we can use it as a bridge between these two spaces. The regular expression that defines it is described in the table 4.8

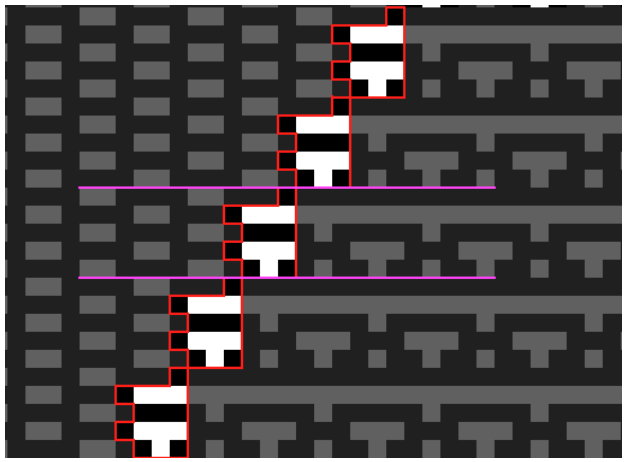


Figura 4.10: Glider 6(g_6) moving within the space of evolutions

Line	Cells	Phase
1	1	1
1000	4	2
111	3	3
1000	4	4
101	3	5

Cuadro 4.8: Lines of cells to form the glider 6

4.4.3. Still-life

Still-life are similar to gliders, with the difference that they do not move to one side, but stay in one position and move exclusively to "down". Within rule 126 with memory we find two stationary structures that live in different periodic backgrounds. Stationary structures will be denoted by an "s" followed by the structure number.

Still-life s_1

The still-life 1 has the shape shown in the figure 4.11

As we can see, it is a very small structure that only occupies 2 evolutions and does not move to any side, so we can describe the cells that compose it in the table 4.9

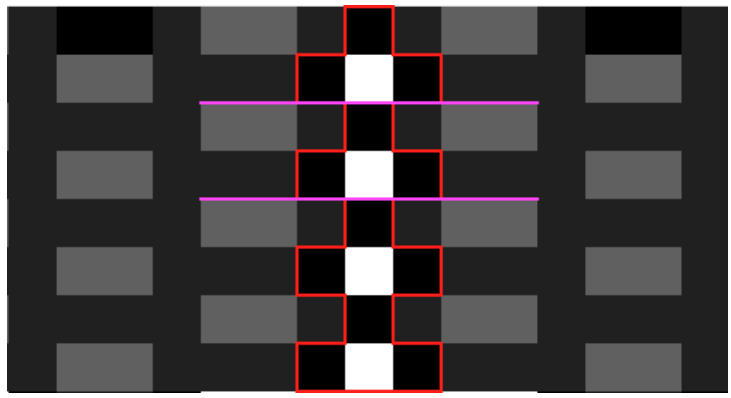


Figura 4.11: Still-life (s_1) moving within the space of evolutions

Line	Cells	Phase
1	1	1
101	3	2

Cuadro 4.9: Cell lines to form the still-life 1

Still-life s_2

The still-life 2 has the form shown in figure 4.12

As we can see, this stationary structure lives in the periodic background 2, but in some evolutions it seems that it disappears. The table that describes the cells that form it is the table 4.10

As we can see in the table, this stationary structure is composed of 10 evolutions and has 3 different phases, which are repeated as our stationary structure progresses.

4.4.4. Gliders-gun

The gliders-gun are "triggers.ºf gliders, we can also understand them as generators of gliders. They are observed in the space of evolutions generating gliders that move in space. Within the 126 rule with memory we can observe a variety of gliders that do not differ much in the gliders they generate, but in their structure to generate them, so some of them will be described. From this moment to refer to a glider we will do it with the word "gun" followed by the glider number.

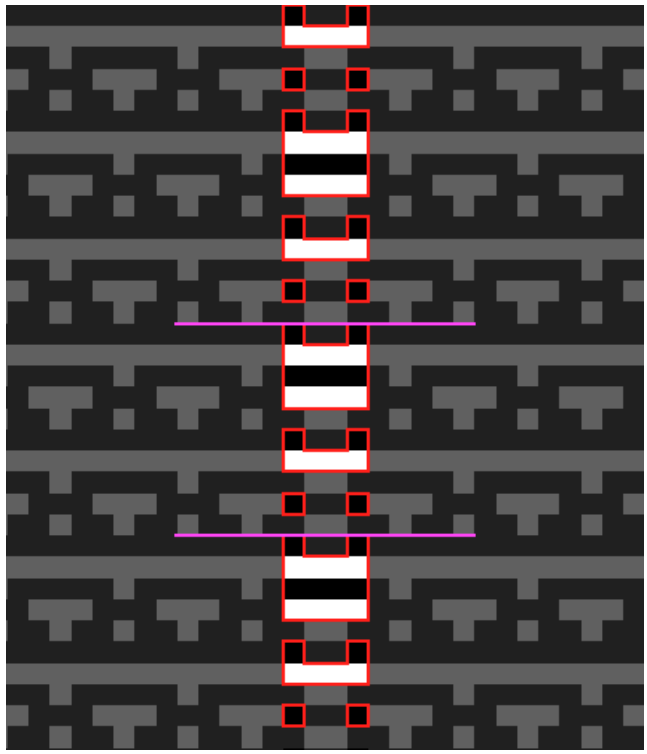


Figura 4.12: Still-life 2(s_2) moving within the space of evolutions

Glider gun 1

The glider-gun 1 is shown in figure 4.13

As we can see, glider gun 1 generates gliders g1 and g2. We can see that within this glider-gun, the generated gliders are separated by the periodic background 1, and we can see that the distance between the gliders is minimal.

Glider gun 2

The glider-gun 2 is shown in figure 4.14

As we can see, the glider-gun 2 generates the gliders g1 and g2. Like the gun1, this glider-gun also lives in the periodic background 1 but the distance between the gliders is greater.

Line	Cells	Phase
1 1	2	1
0000	4	2
1111	4	3
0000	4	4
	0	5
1 1	2	6
0000	4	7
	0	8
1 1	2	9
	0	10

Cuadro 4.10: Cell lines to form the still-life 2

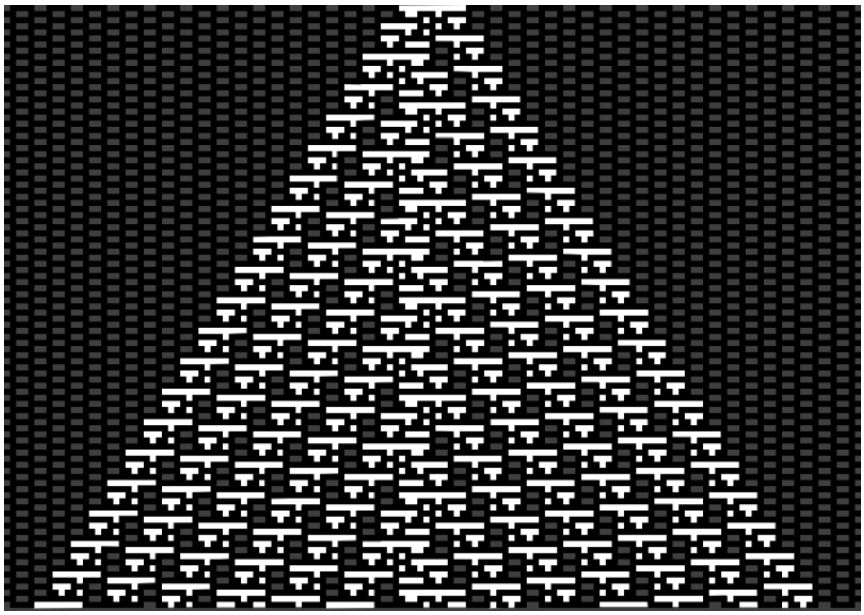


Figura 4.13: Glider-gun 1 (gun1) generating the gliders g1 and g2 within the space of evolutions

4.5. Points of collision of the elements within the space of evolutions

The Collision points are evolutions where a glider or a stationary structure collide with each other in some evolution and generate a new pattern. Within the gliders of

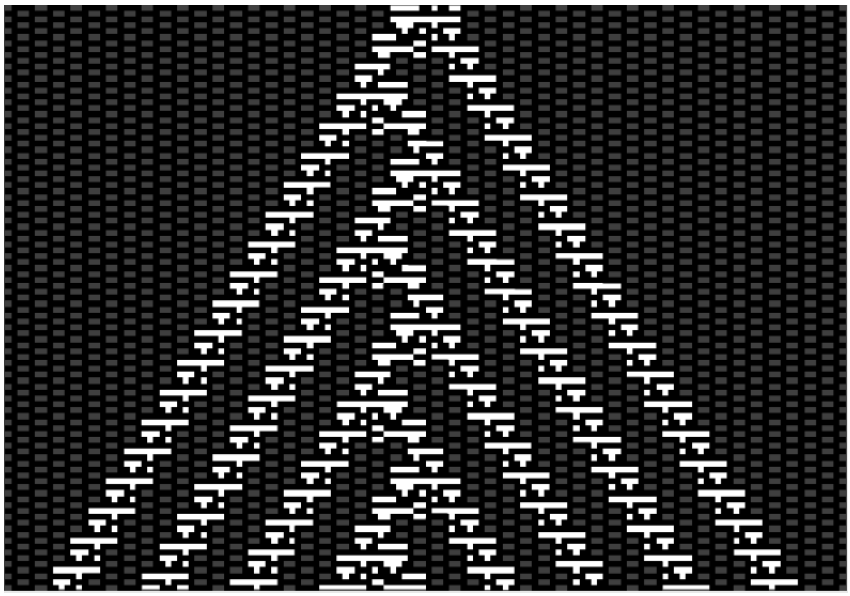


Figura 4.14: Glider-gun 2 (gun2) generating the gliders g1 and g2 within the space of evolutions

rule 126 with memory we find that they vary in number of points depending on the type of glider, the same happens with still-life. In this section the various points for the gliders g1, g2 and the still-life s1 and s2 will be shown.

4.5.1. Collision points of the gliders

Points on the glider g_1

The collision points of the glider g_1 are those that are indicated with arrows in the figure 4.15

As we can see, the glider g_1 has 3 collision points, which we can vary by rotating it in its 5 different phases.

Points on the glider g_2

The collision points of the glider g_2 are those that are indicated with arrows in the figure 4.16

As we can see, the glider g_2 has 3 collision points, which we can vary by rotating it in its 5 different phases.

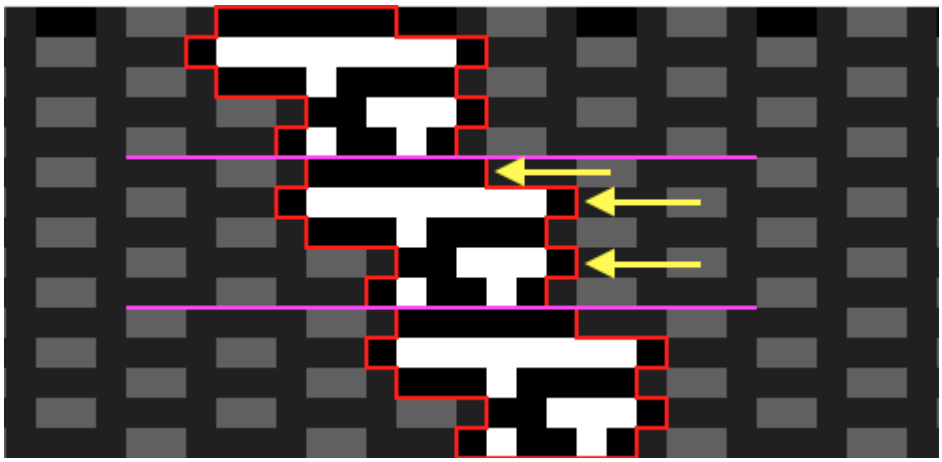


Figura 4.15: The collision points of the glider g_1 in the space of evolutions

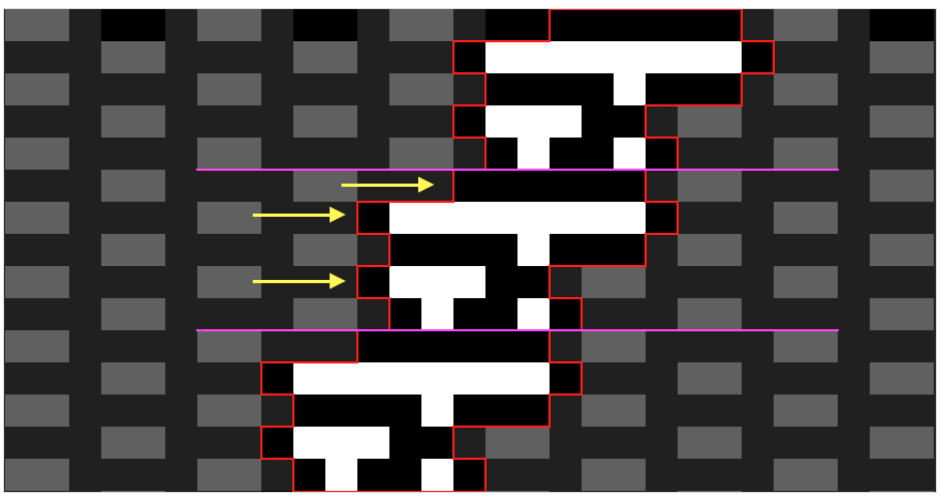


Figura 4.16: The collision points of the glider g_2 in the space of evolutions

Points on the glider g_3

The collision points of the glider g_3 are those that are indicated with arrows in the figure 4.17

As we can see, the glider g_3 has only one collision point.

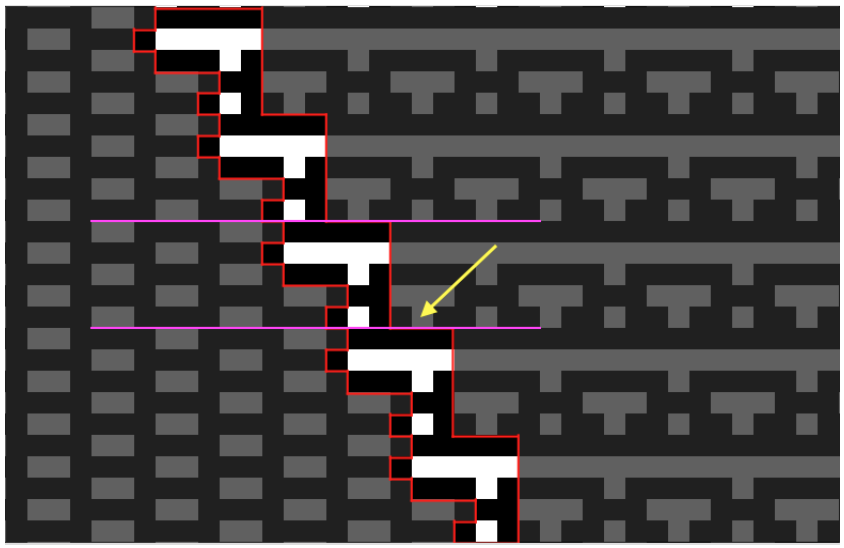


Figura 4.17: The collision points of the glider g_3 in the space of evolutions

Points on the glider g_4

The collision points of the glider g_4 are those that are indicated with arrows in the figure 4.18

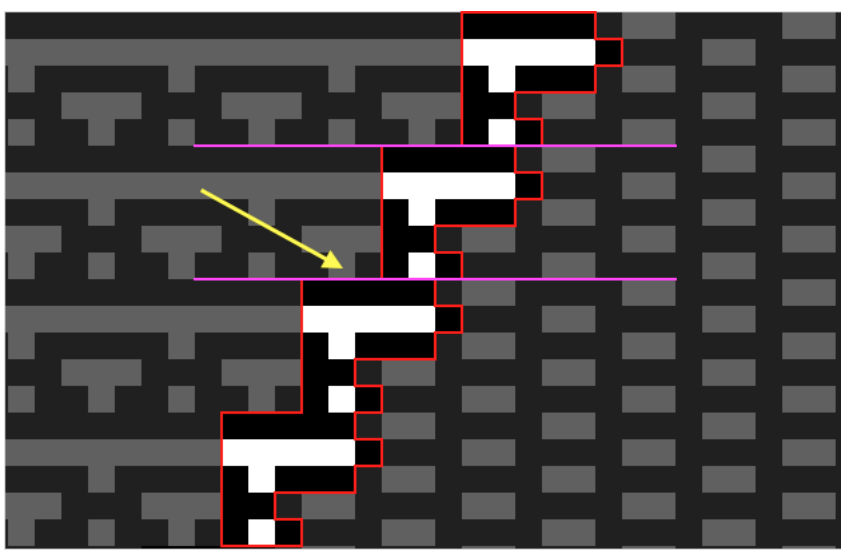


Figura 4.18: The collision points of the glider g_4 in the space of evolutions

As we can see, the glider g_4 has only one collision point.

Points on the glider g_5

The collision points of the glider g_5 are those that are indicated with arrows in the figure 4.19

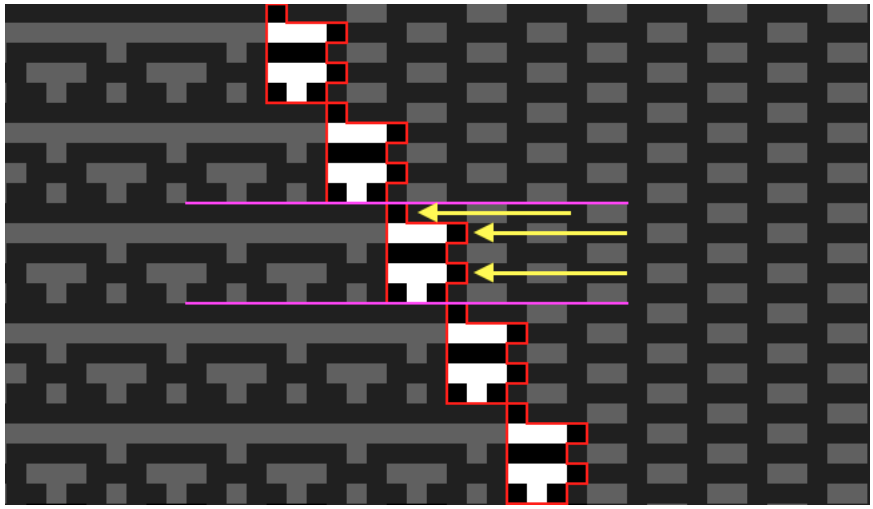


Figura 4.19: The collision points of the glider g_5 in the space of evolutions

As we can see, the glider g_5 has 3 collision points, which we can vary by rotating it in its 5 different phases.

Puntos en el glider g_6

The collision points of the glider g_6 are those that are indicated with arrows in the figure 4.20

As we can see, the glider g_6 has 3 collision points, which we can vary by rotating it in its 5 different phases.

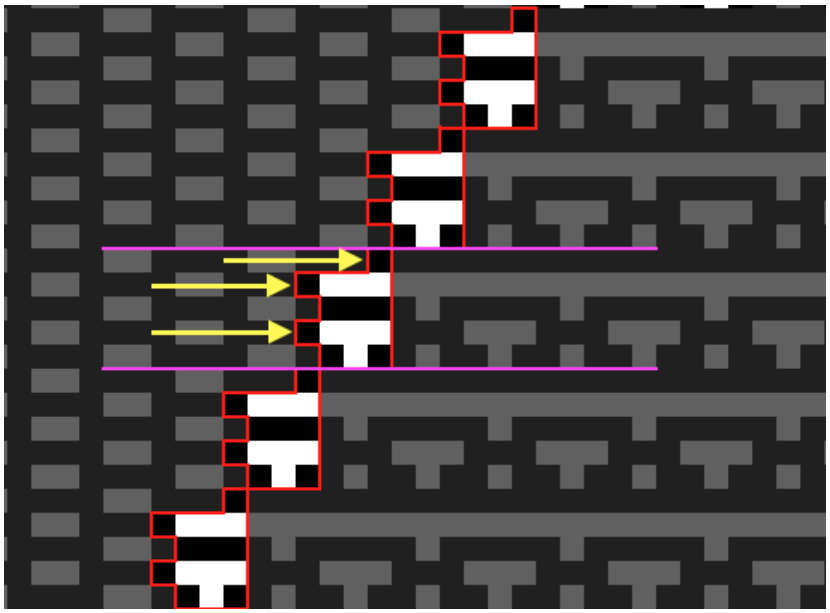


Figura 4.20: The collision points of the glider g_6 in the space of evolutions

4.5.2. Collision points of the still-life

Points in the still-life s_1

The still-life s_1 has the collision points indicated with arrows in the figure 4.21

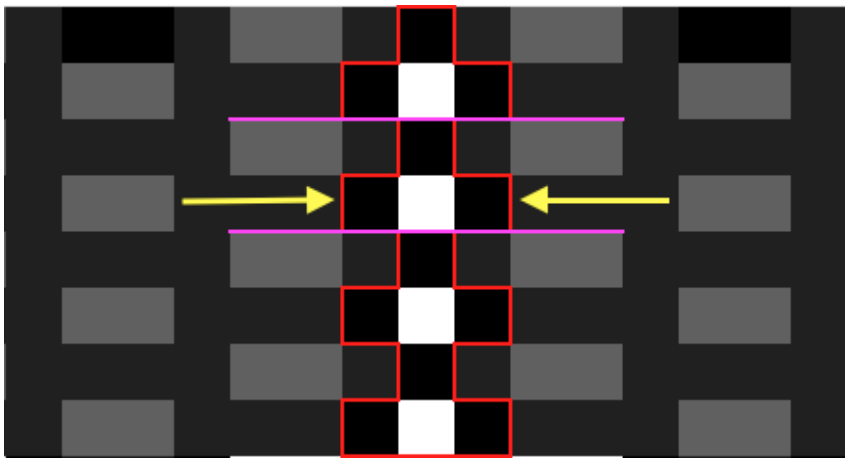


Figura 4.21: Collision points of the still-life s_1 in the space of evolutions

As we can see in the figure 4.21, this still-life s_1 has two collision points on the

sides.

Points in the still-life s_2

The still-life s_2 has the collision points indicated with arrows in the figure 4.22

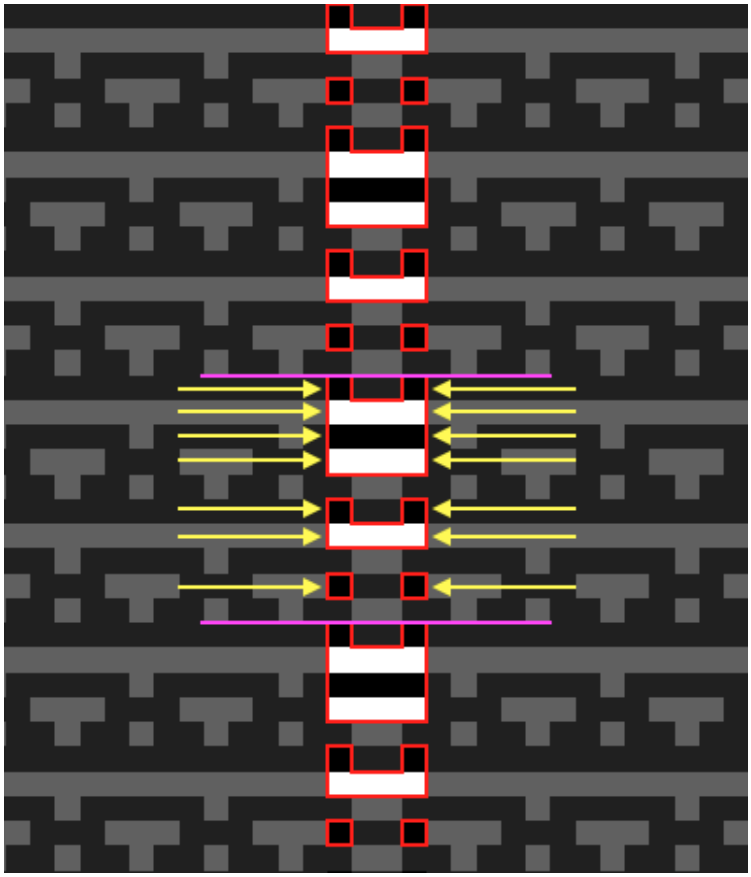


Figura 4.22: Collision points of the still-life s_2 in the space of evolutions

As we can see in the figure 4.22, this still-life s_2 has 14 collision points, 7 on each side.

4.6. Collisions between the elements within the evolution space

In this section we will summarize the different collisions obtained with the identified structures within the evolution space, to see all the collisions carried out exhaustively see the appendix B. These collisions were classified according to the number of particles that intervened in the collision, being those of two particles binary collisions, those of three ternary particles and so on.

To understand the result generated by the collisions, we will use a notation in the form of an equation in the following way:

$g_1 \leftrightarrow s_2 \leftrightarrow g_3 \leftrightarrow ... \leftrightarrow g_n \leftrightarrow s_n = K$

Where:

1. g_n will denote the glider with its respective number.

2. s_n will denote the still-life with its respective number.

3. K will denote a combination of gliders, tiles for filtering, etc, depending on the result of the collision. K can own particles separated by:

   - **Raised to some power:** It indicates that they are equal particles and that they are not separated, the number of the power indicates the amount of particles that are together. Examples: g_1^2, g_3^4, etc.
   - **Multiplied:** For equal particles it indicates that they are separated by a tile for filtering, the number that comes with them indicates the quantity of particles. For different particles it indicates that they are stuck together. Examples: $g_1 g_2$, $2g_1$, etc.
   - **Sign of +:** For equal particles it indicates that they are separated by more than one tile for filtering. For different particles it indicates that they are separated by one or more tiles for filtering. Examples: $g_1 + g_1$, $g_3 + g_4$, etc.

4.6.1. Binary collisions

We will begin with collisions that only involve two elements within the space of evolutions: the Binary collisions.

Gliders: g_1 and g_2

In the figure 4.23 we can observe the collisions obtained with the gliders g_1 y g_2 respectively, below each image is the equation that describes them.

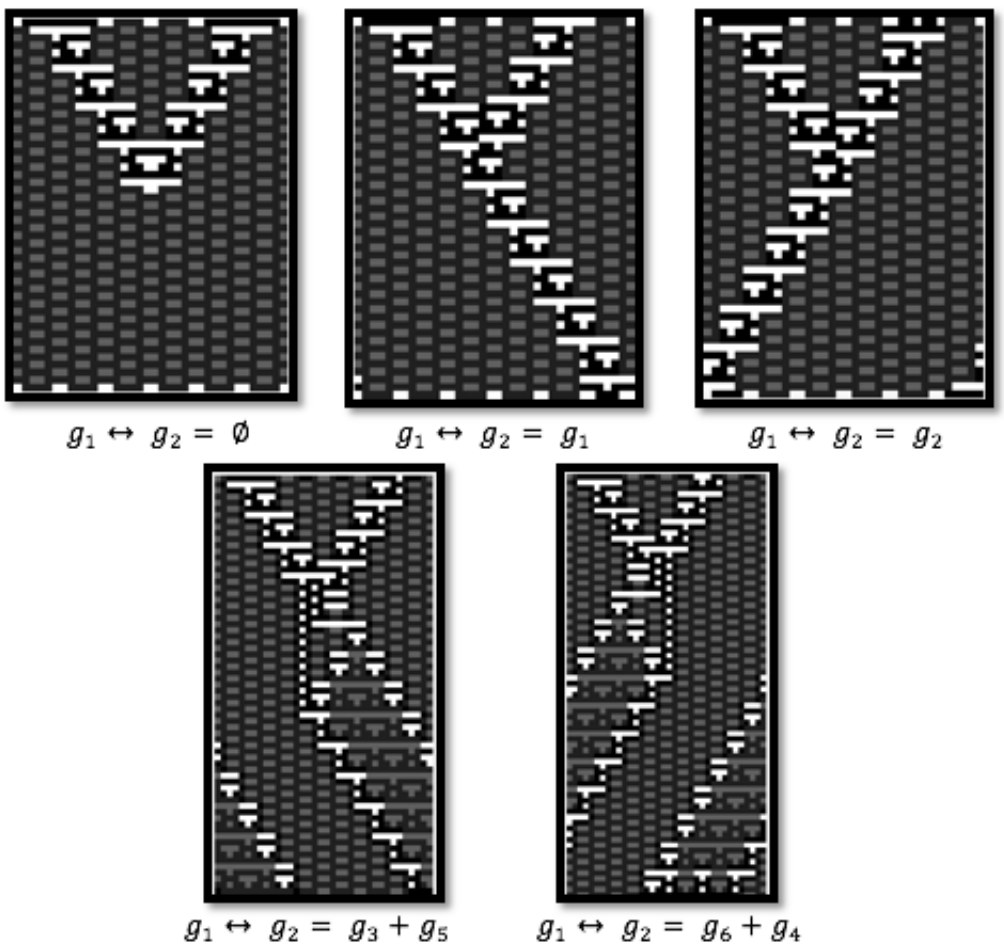


Figura 4.23: The different results obtained from colliding g_1 and g_2 with its respective equation

In the figure 4.23 we can observe 5 different collisions that produce different

results, the equations indicated are:

1. $g_1 \leftrightarrow g_2 = \emptyset$
2. $g_1 \leftrightarrow g_2 = g_1$
3. $g_1 \leftrightarrow g_2 = g_2$
4. $g_1 \leftrightarrow g_2 = g_3 + g_5$
5. $g_1 \leftrightarrow g_2 = g_6 + g_4$

Gliders: g_1 and g_6

In the figure 4.24 we can observe the collisions obtained with the gliders g_1 y g_6 respectively, below each image is the equation that describes them.

In the figure 4.24 we can observe 5 different collisions that produce different results, the only collision that produces g_6 again it is indicated twice because on one occasion it arrives with a delay, while on the other it continues its way normally. The equations indicated are:

1. $g_1 \leftrightarrow g_6 = g_6 - with - delay$
2. $g_1 \leftrightarrow g_6 = g_3^2$
3. $g_1 \leftrightarrow g_6 = g_6$
4. $g_1 \leftrightarrow g_6 = g_2 + g_6 + g_5 + g_3$
5. $g_1 \leftrightarrow g_6 = g_6 + g_5 + g_3$

Gliders: g_1 and s_1

In the figure 4.25 we can observe the collision obtained with the glider g_1 and the still-life s_1 respectively.

In the figure 4.25 we can see that the collision obtained is simply that the still-life s_1 is maintained while the glider g_1 seems that it is destroyed, so we rotate the glider we will obtain the same result, so the list of collisions is simply this:

1. $g_1 \leftrightarrow s_1 = s_1$

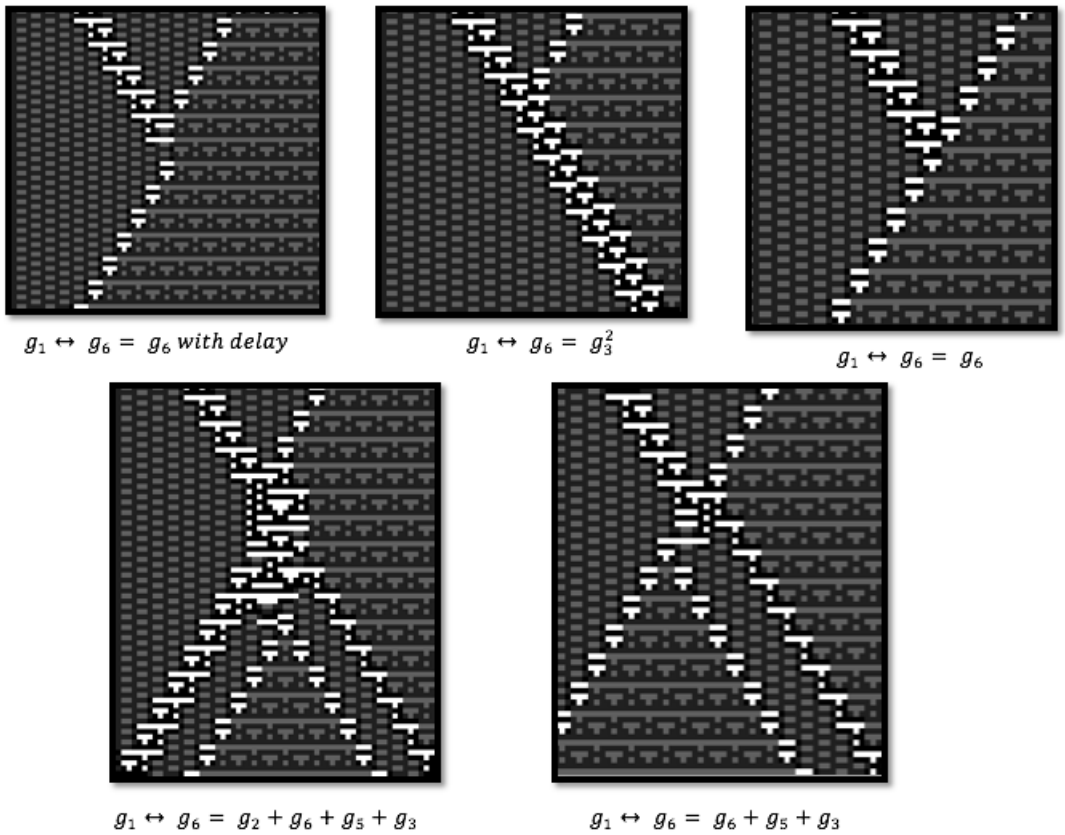
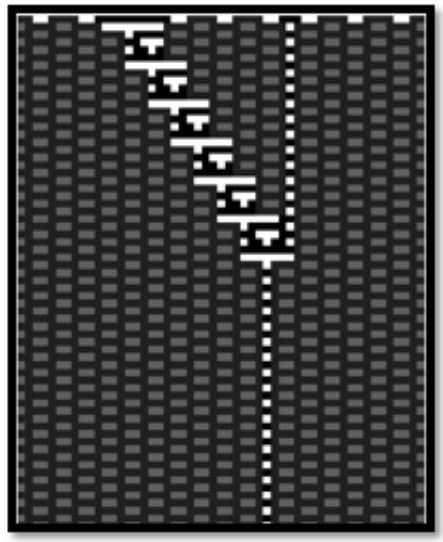


Figura 4.24: The different results obtained from colliding g_1 and g_6 with its respective equation



$g_1 \leftrightarrow s_1 = s_1$

Figura 4.25: The collision obtained between g_1 and s_1

Gliders: g_3 and g_4

In the figure 4.26 we can observe the collision obtained with the glider g_3 and the glider g_4 respectively.

In the figure 4.26 we can see that the collision obtained generates a new particle that did not even participate in the collision (s_1). The equation that describes it is the following:

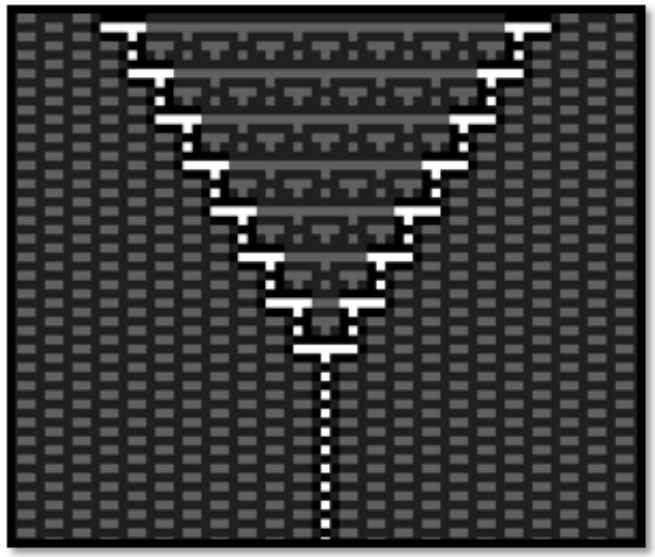
1. $g_3 \leftrightarrow g_4 = s_1$

Gliders: g_3 and s_2

In the figure 4.27 we can observe the collision obtained with the glider g_3 y el still-life s_2 respectivamente.

In the figure 4.27 we can see that the collision obtained generates a new particle that did not even participate in the collision (g_6). The equation that describes it is the following.

1. $g_3 \leftrightarrow s_2 = g_6$



$g_3 \leftrightarrow g_4 = s_1$

Figura 4.26: The collision obtained between g_3 and g_4

Gliders: g_5 and g_2

In the figure 4.28 we can observe the collisions obtained with the glider g_5 and the glider g_2 respectively.

As we can see in the figure 4.28, two collisions keep g_5 but one causes a delay and displaces it, while the other is not altered, the other collisions generate new particles. The equations that define the collisions obtained are:

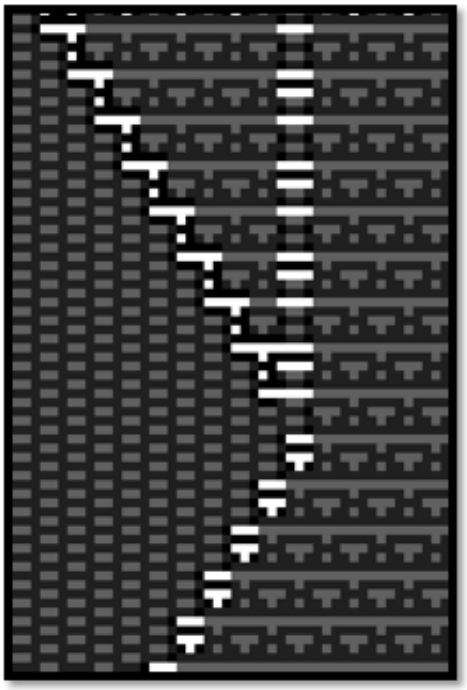
1. $g_5 \leftrightarrow g_2 = g_5 - with - delay$

2. $g_5 \leftrightarrow g_2 = g_5$

3. $g_5 \leftrightarrow g_2 = g_6 + g_5$

4. $g_5 \leftrightarrow g_2 = g_4 + g_6 + g_5 + g_1$

5. $g_5 \leftrightarrow g_2 = g_3^2$



$g_3 \leftrightarrow s_2 = g_6$

Figura 4.27: The collision obtained between g_3 and s_2

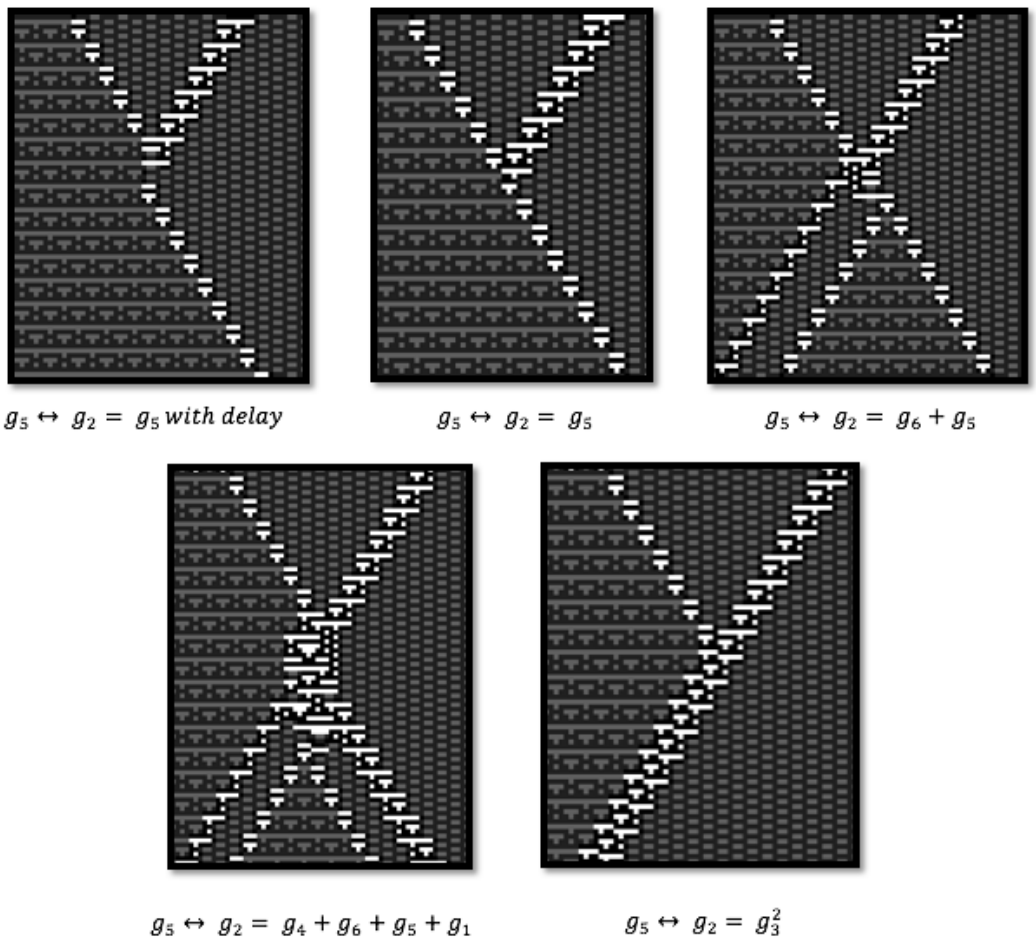


Figura 4.28: The collision obtained between g_5 and g_2

Gliders: g_5 and g_6

In the figure 4.29 we can observe the collisions obtained with the glider g_5 and the glider g_6 respectively.

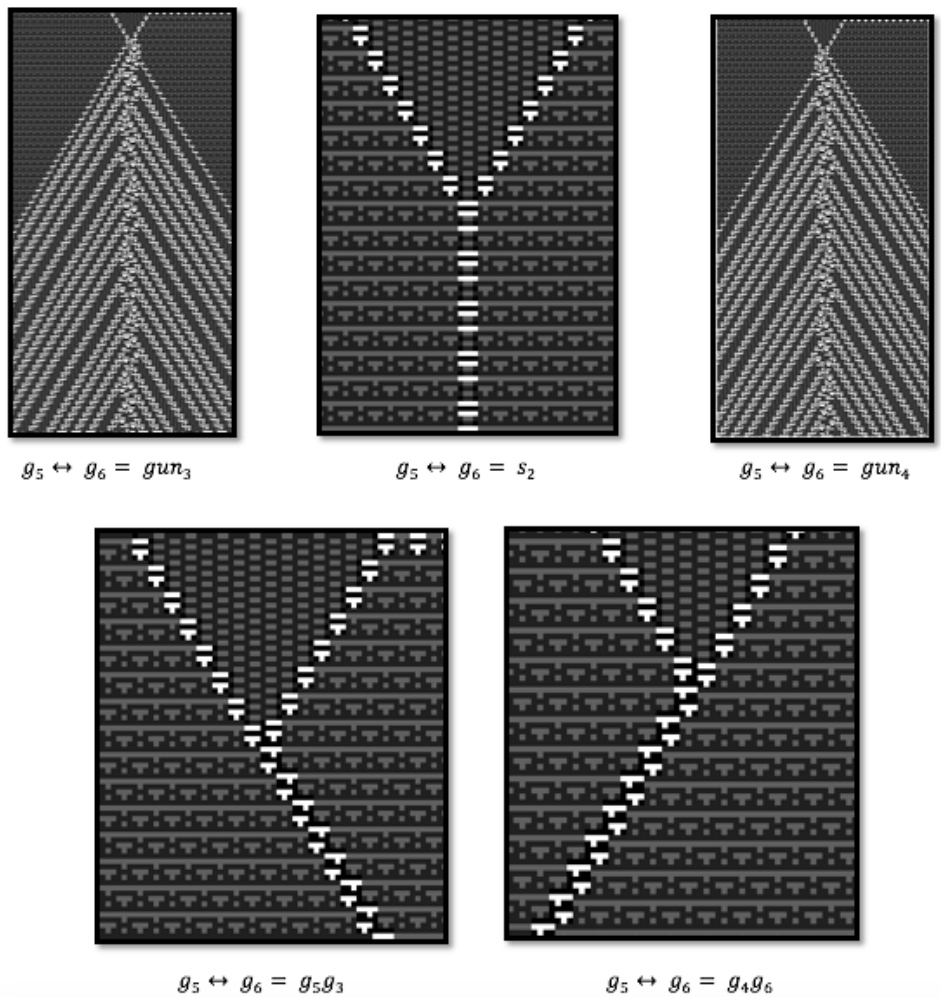


Figura 4.29: The collision obtained between g_5 and g_6

As we can see in the figure 4.29, we get a glider-gun from the collisions, this facilitates its creation. The equations that define the collisions obtained are:

1. $g_5 \leftrightarrow g_6 = gun_3$

2. $g_5 \leftrightarrow g_6 = s_2$

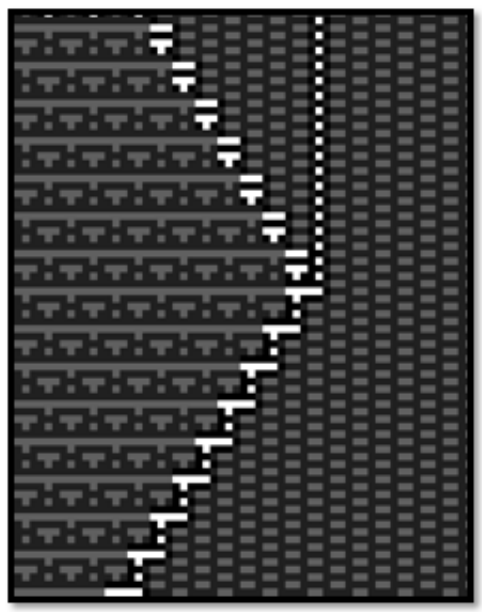
3. $g_5 \leftrightarrow g_6 = gun_4$

4. $g_5 \leftrightarrow g_6 = g_5 g_3$

5. $g_5 \leftrightarrow g_6 = g_4 g_6$

Gliders: g_5 and s_1

In the figure 4.30 we can observe the collision obtained with the glider g_5 and the still-life s_1 respectively.



$g_5 \leftrightarrow s_1 = g_4$

Figura 4.30: The collision obtained between g_5 and s_1

As we can see in the figure 4.30, it seems that the glider bounces, causing it to mutate g_4, the equation that defines the interaction is the following:

1. $g_5 \leftrightarrow s_1 = g_4$

Gliders: s_1 and g_2

In the figure 4.31 we can observe the collision obtained with the still-life s_1 and the glider g_2 respectively.

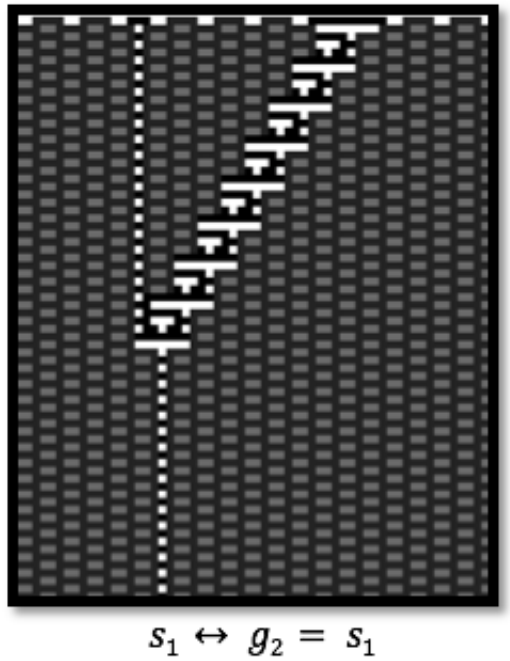


Figura 4.31: The collision obtained between s_1 and g_2

As we can see in the figure 4.31, absorbs the glider and continues s_1 a little displaced. The equation that defines the interaction is the following:

1. $s_1 \leftrightarrow g_2 = s_1$

Gliders: s_1 and g_6

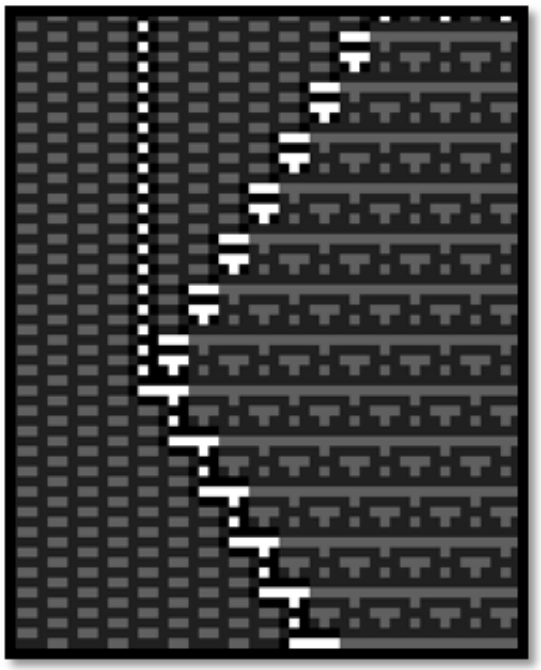
In the figure 4.32 we can observe the collision obtained with the still-life s_1 and the glider g_6 respectively.

As we can see in the figure 4.32, it bounces off the glider and mutates to g_3. The equation that defines the interaction is the following:

1. $s_1 \leftrightarrow g_6 = g_3$

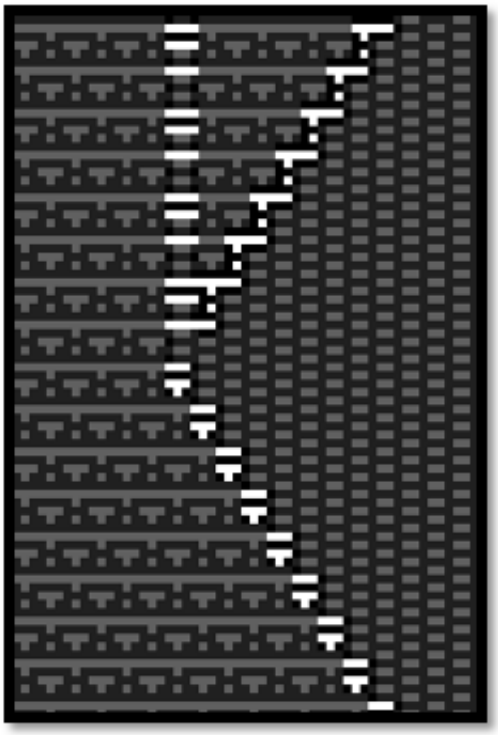
Gliders: s_2 and g_4

In the figure 4.33 we can observe the collision obtained with the still-life s_2 and the glider g_4 respectively.



$s_1 \leftrightarrow g_6 = g_3$

Figura 4.32: The collision obtained between s_1 and g_6



$s_2 \leftrightarrow g_4 = g_5$

Figura 4.33: The collision obtained between s_2 and g_4

As we can see in the figure 4.33, it bounces off the glider and mutates to g_5. The equation that defines the interaction is the following:

1. $s_2 \leftrightarrow g_4 = g_5$

Summary of binary collisions

In the table 4.11, all the equations of the collisions that we exemplify are summarized. In total there are 165 possible binary collisions, this was calculated using set theory (more details in the appendix B).

Equation	**Result**
$g_1 \leftrightarrow g_2$	$\emptyset$
$g_1 \leftrightarrow g_2$	g_1
$g_1 \leftrightarrow g_2$	g_2
$g_1 \leftrightarrow g_2$	$g_3 + g_5$
$g_1 \leftrightarrow g_2$	$g_6 + g_4$
$g_1 \leftrightarrow g_6$	$g_6 - with - delay$
$g_1 \leftrightarrow g_6$	g_3^2
$g_1 \leftrightarrow g_6$	g_6
$g_1 \leftrightarrow g_6$	$g_2 + g_6 + g_5 + g_3$
$g_1 \leftrightarrow g_6$	$g_6 + g_5 + g_3$
$g_1 \leftrightarrow s_1$	s_1
$g_3 \leftrightarrow g_4$	s_1
$g_3 \leftrightarrow s_2$	g_6
$g_5 \leftrightarrow g_2$	$g_5 - with - delay$
$g_5 \leftrightarrow g_2$	g_5
$g_5 \leftrightarrow g_2$	$g_6 + g_5$
$g_5 \leftrightarrow g_2$	$g_4 + g_6 + g_5 + g_1$
$g_5 \leftrightarrow g_2$	g_3^2
$g_5 \leftrightarrow g_6$	gun_3
$g_5 \leftrightarrow g_6$	s_2
$g_5 \leftrightarrow g_6$	gun_4

Continued on the next page

Cuadro 4.11 – *Continued from previous page*

Equation	**Result**
$g_5 \leftrightarrow g_6$	$g_5 g_3$
$g_5 \leftrightarrow g_6$	$g_4 g_6$
$g_5 \leftrightarrow s_1$	g_4
$s_1 \leftrightarrow g_2$	s_1
$s_1 \leftrightarrow g_6$	g_3
$s_2 \leftrightarrow g_4$	g_5

Cuadro 4.11: Summary table of binary collisions

4.6.2. Ternary collisions

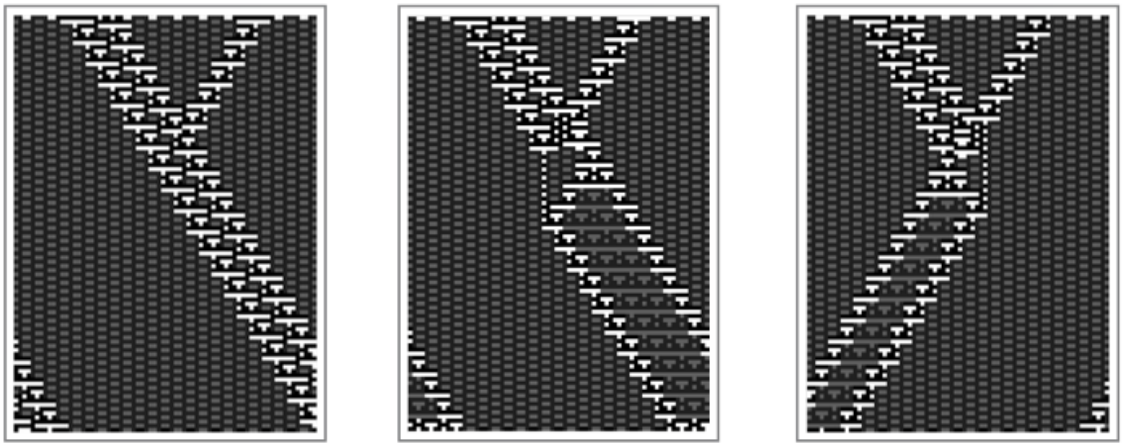
In this section we will describe the collisions that involve 3 structures within the evolutions space: Ternary collisions.

Gliders: $2g_1$ **and** g_2

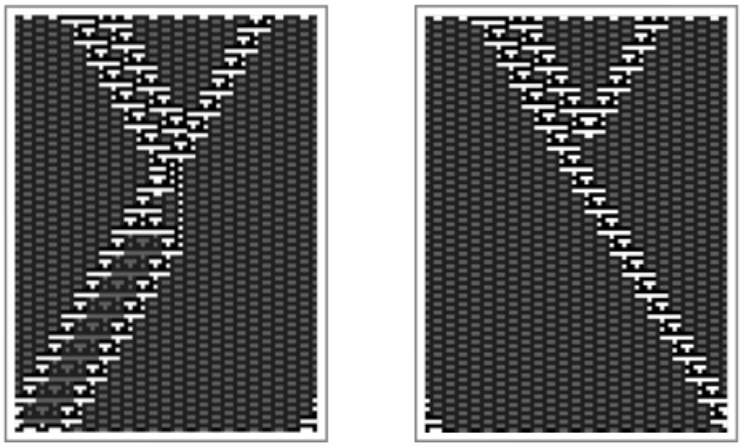
In the figure 4.34 we can observe the collisions obtained with the gliders $2g_1$ y g_2 respectively, below each image is the equation that describes them.

In the figure 4.34 we can observe 5 different collisions that produce different results, the equations indicated are:

1. $2g_1 \leftrightarrow g_2 = 2g_1$
2. $2g_1 \leftrightarrow g_2 = g_3 + g_5$
3. $2g_1 \leftrightarrow g_2 = g_6 + g_4$
4. $2g_1 \leftrightarrow g_2 = g_6 + g_4$
5. $2g_1 \leftrightarrow g_2 = g_1$



$2g_1 \leftrightarrow g2 = 2g_1$ $2g_1 \leftrightarrow g2 = g_3 + g_5$ $2g_1 \leftrightarrow g2 = g_6 + g_4$



$2g_1 \leftrightarrow g2 = g_6 + g_4$ $2g_1 \leftrightarrow g2 = g_1$

Figura 4.34: The different results obtained from colliding $2g_1$ and g_2 with its respective equation

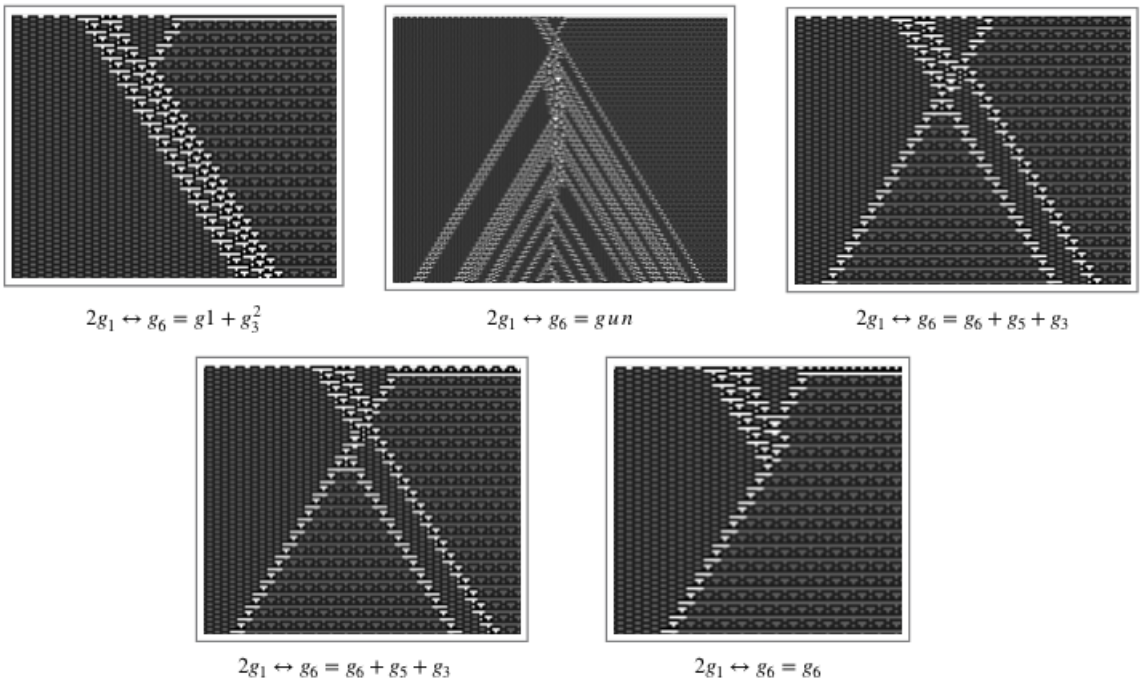


Figura 4.35: The different results obtained from colliding $2g_1$ and g_6 with its respective equation

Gliders: $2g_1$ **and** g_6

In the figure 4.35 we can observe the collisions obtained with the gliders $2g_1$ y g_6 respectively, below each image is the equation that describes them.

In the figure 4.35 we can observe 5 different collisions that produce different results, the equations indicated are:

1. $2g_1 \leftrightarrow g_6 = g_1 + g_3^2$
2. $2g_1 \leftrightarrow g_6 = gun$
3. $2g_1 \leftrightarrow g_6 = g_6 + g_5 + g_3$
4. $2g_1 \leftrightarrow g_6 = g_6 + g_5 + g_3$
5. $2g_1 \leftrightarrow g_6 = g_6$

Gliders: $2g_1$ **and** s_1

In the figure 4.36 we can observe the collisions obtained with the gliders $2g_1$ and the still-life s_1 respectively, below each image is the equation that describes them.

In the figure 4.36 we can only observe a different collision, which equation is:

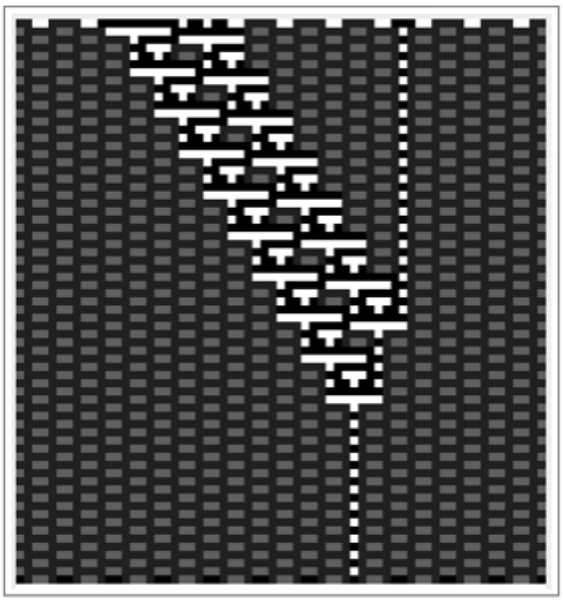
1. $2g_1 \leftrightarrow s_1 = s_1$

Gliders: $g_1 + g_1$ **and** g_2

In the figure 4.37 we can observe the collisions obtained with the gliders $g_1 + g_1$ y g_2 respectively, below each image is the equation that describes them.

In the figure 4.37 we can observe 5 different collisions that produce different results, the equations indicated are:

1. $g_1 + g_1 \leftrightarrow g_2 = \emptyset$
2. $g_1 + g_1 \leftrightarrow g_2 = g_1$
3. $g_1 + g_1 \leftrightarrow g_2 = g_1 + g_1$
4. $g_1 + g_1 \leftrightarrow g_2 = g_3 + g_5$
5. $g_1 + g_1 \leftrightarrow g_2 = g_2$



$2g_1 \leftrightarrow s_1 = s_1$

Figura 4.36: The different results obtained from colliding $2g_1$ and s_1 with its respective equation

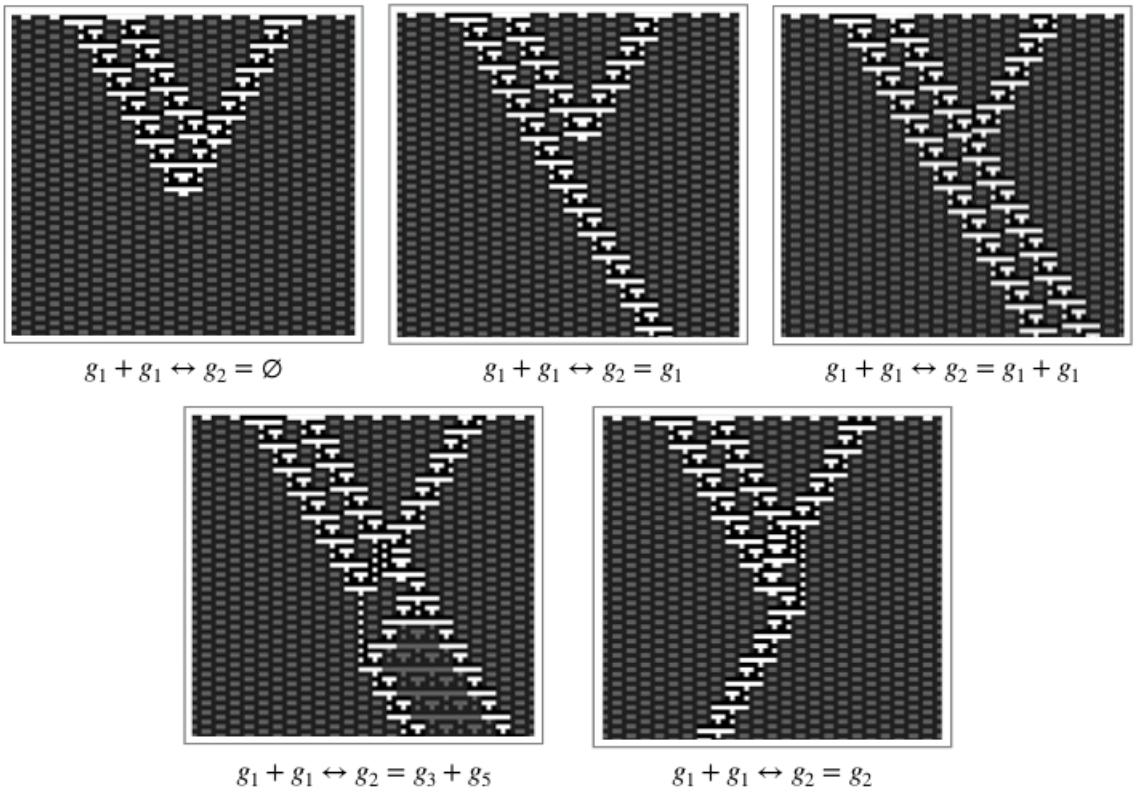


Figura 4.37: The different results obtained from colliding $g_1 + g_1$ and g_2 with its respective equation

Gliders: $g_1 + g_1$ **and** g_6

In the figure 4.38 we can observe the collisions obtained with the gliders $g_1 + g_1$ y g_6 respectively, below each image is the equation that describes them.

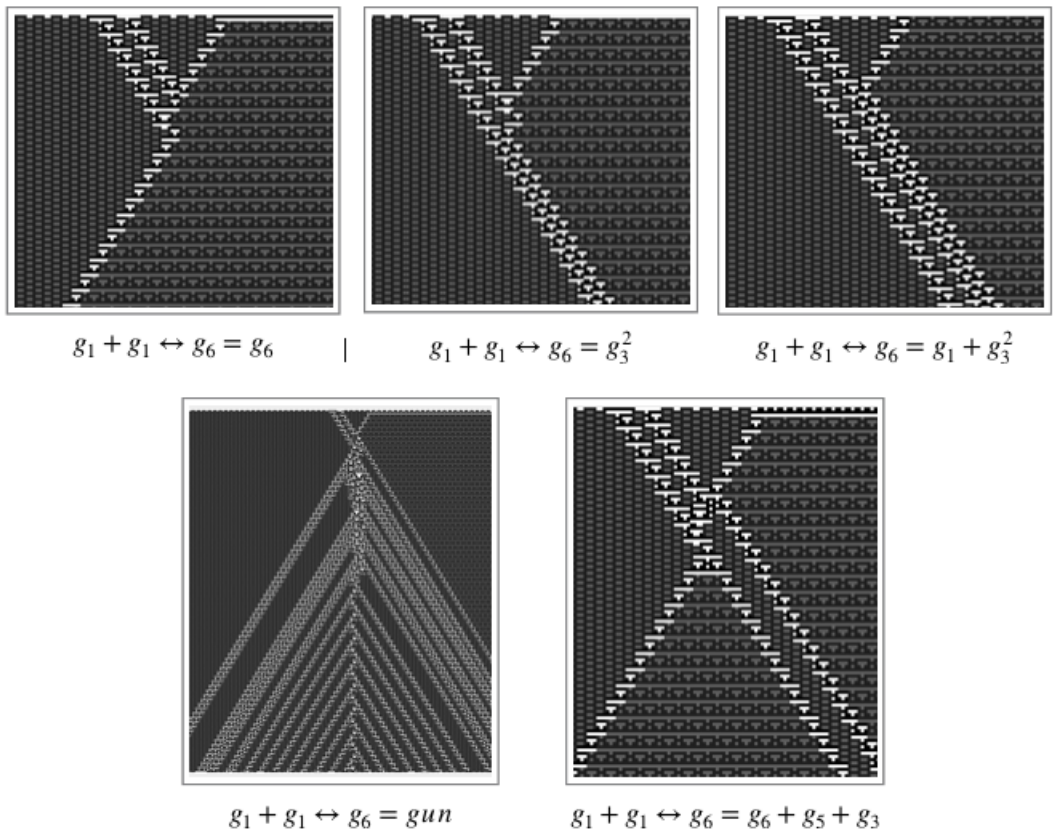


Figura 4.38: The different results obtained from colliding $g_1 + g_1$ and g_6 with its respective equation

In the figure 4.38 we can observe 5 different collisions that produce different results, the equations indicated are:

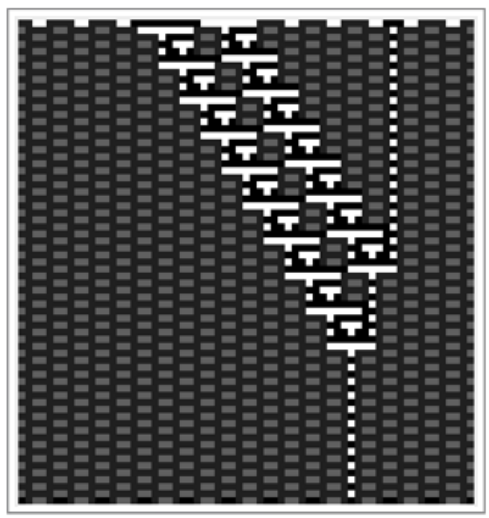
1. $g_1 + g_1 \leftrightarrow g_6 = g_6$
2. $g_1 + g_1 \leftrightarrow g_6 = g_3^2$
3. $g_1 + g_1 \leftrightarrow g_6 = g_1 + g_3^2$

4. $g_1 + g_1 \leftrightarrow g_6 = gun$

5. $g_1 + g_1 \leftrightarrow g_6 = g_6 + g_5 + g_3$

Gliders: $g_1 + g_1$ **and** s_1

In the figure 4.39 we can observe the collisions obtained with the gliders $g_1 + g_1$ and the still-life s_1 respectively, below each image is the equation that describes them.



$g_1 + g_1 \leftrightarrow s_1 = s_1$

Figura 4.39: The different results obtained from colliding $g_1 + g_1$ and s_1 with its respective equation

In the figure 4.39 we can only observe a different collision, which equation is:

1. $g_1 + g_1 \leftrightarrow s_1 = s_1$

Gliders: g_1 **and** $2g_2$

In the figure 4.40 we can observe the collisions obtained with the gliders g_1 y $2g_2$ respectively, below each image is the equation that describes them.

In the figure 4.40 we can observe 5 different collisions that produce different results, the equations indicated are:

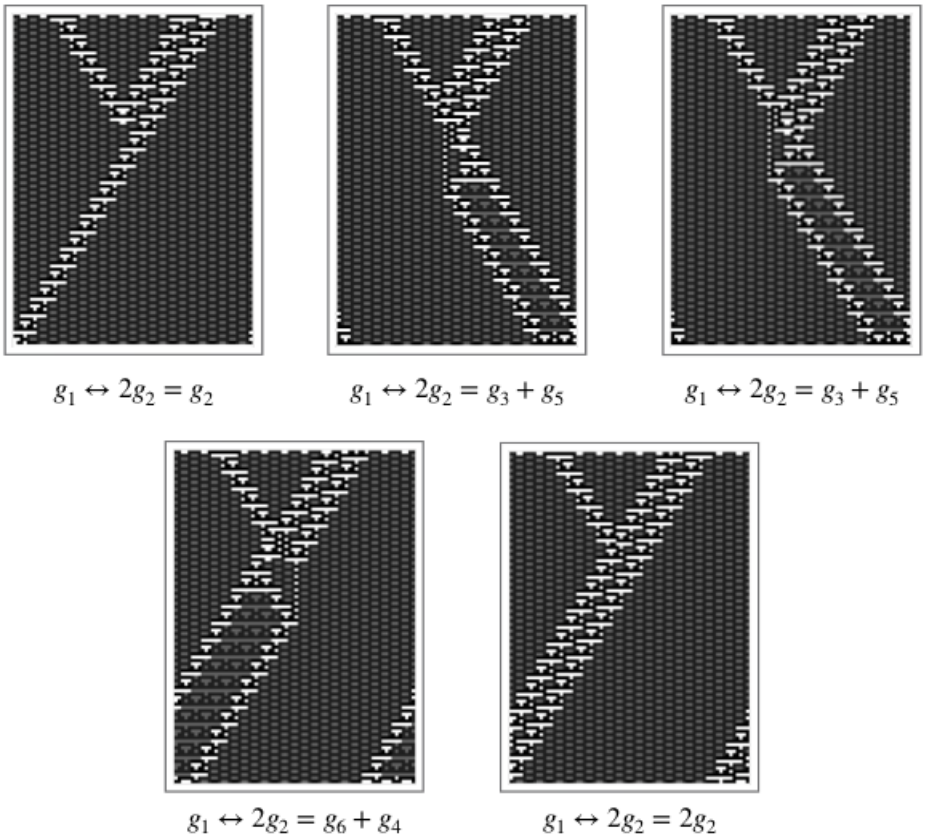
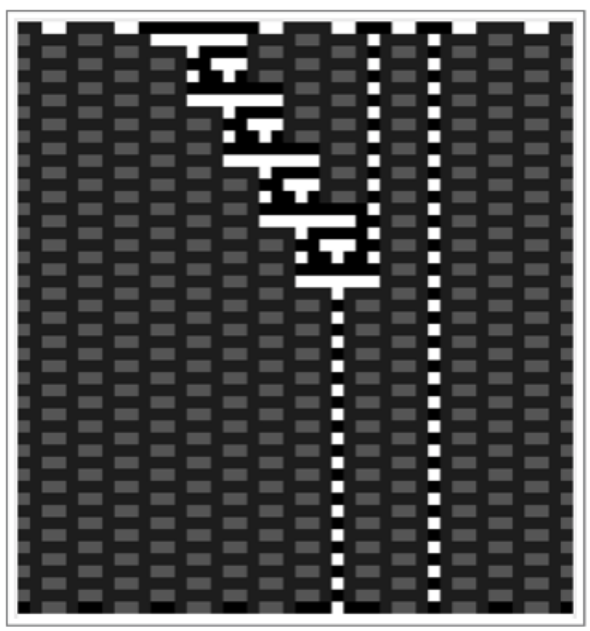


Figura 4.40: The different results obtained from colliding g_1 and $2g_2$ with its respective equation

1. $g_1 \leftrightarrow 2g_2 = g_2$
2. $g_1 \leftrightarrow 2g_2 = g_3 + g_5$
3. $g_1 \leftrightarrow 2g_2 = g_3 + g_5$
4. $g_1 \leftrightarrow 2g_2 = g_6 + g_4$
5. $g_1 \leftrightarrow 2g_2 = 2g_2$

Gliders: g_1 and $2s_1$

In the figure 4.41 we can observe the collisions obtained with the gliders g_1 and the still-life $2s_1$ respectively, below each image is the equation that describes them.



$g_1 \leftrightarrow 2s_1 = s_1 + s_1$

Figura 4.41: The different results obtained from colliding g_1 and $2s_1$ with its respective equation

In the figure 4.41 we can only observe a different collision, which equation is:

1. $g_1 \leftrightarrow 2s_1 = s_1 + s_1$

Gliders: g_1 and $g_2 + g_2$

In the figure 4.42 we can observe the collisions obtained with the gliders g_1 and $g_2 + g_2$ respectively, below each image is the equation that describes them.

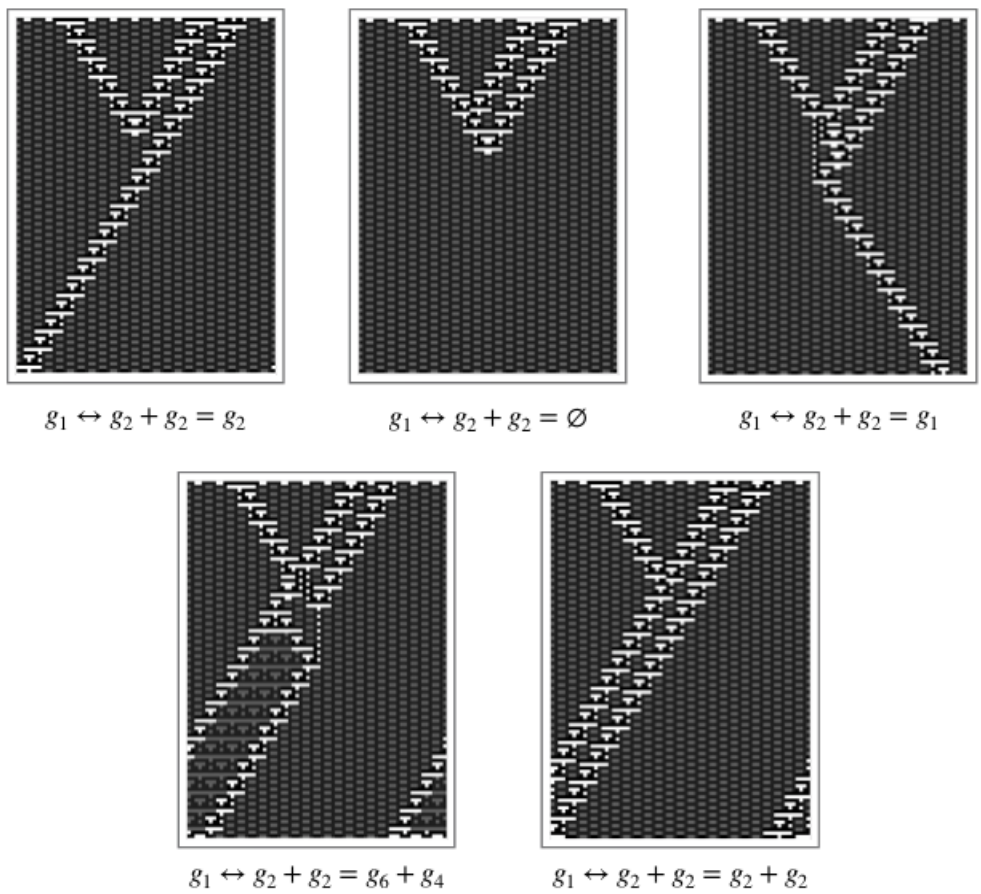


Figura 4.42: The different results obtained from colliding g_1 and $g_2 + g_2$ with its respective equation

In the figure 4.42 we can observe 5 different collisions that produce different results, the equations indicated are:

1. $g_1 \leftrightarrow g_2 + g_2 = g_2$

2. $g_1 \leftrightarrow g_2 + g_2 = \emptyset$

3. $g_1 \leftrightarrow g_2 + g_2 = g_1$

4. $g_1 \leftrightarrow g_2 + g_2 = g_6 + g_4$

5. $g_1 \leftrightarrow g_2 + g_2 = g_2 + g_2$

Gliders: g_1 and g_2^2

In the figure 4.43 we can observe the collisions obtained with the gliders g_1 y g_2^2 respectively, below each image is the equation that describes them.

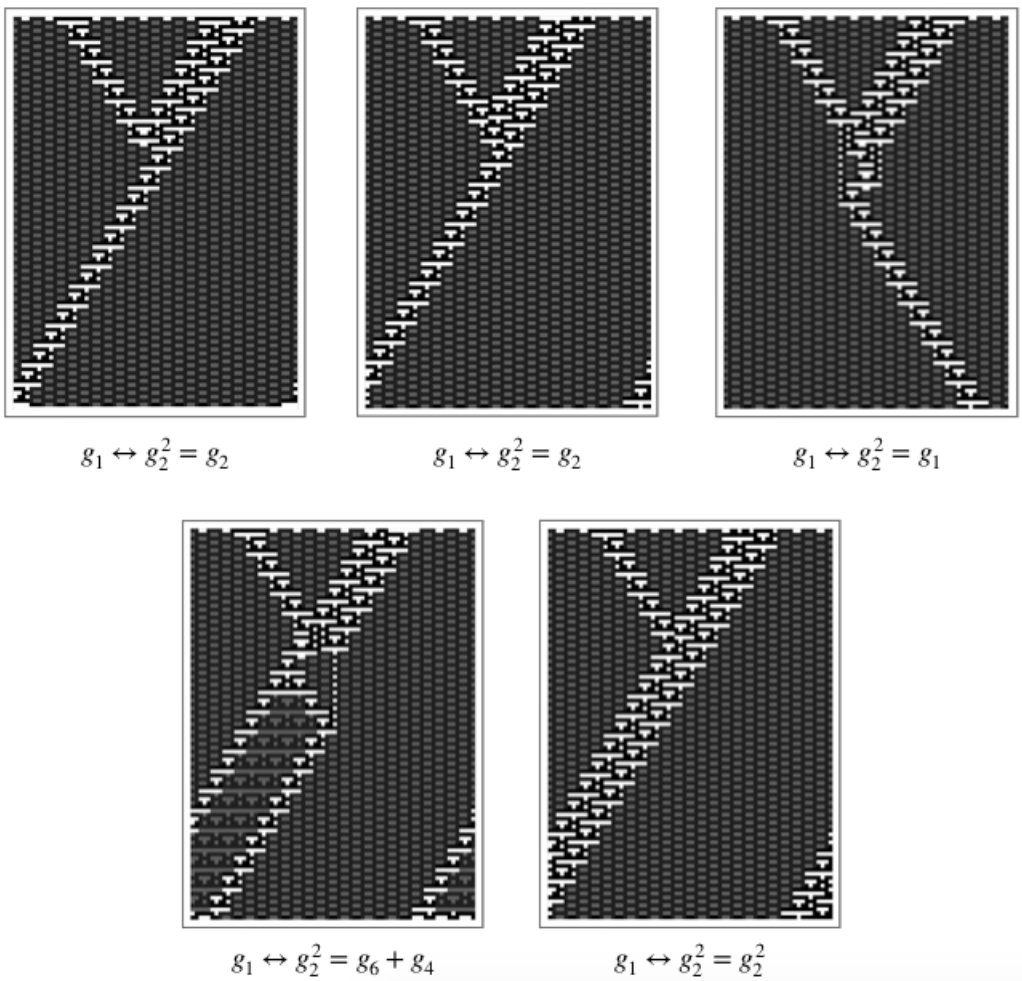


Figura 4.43: The different results obtained from colliding g_1 and g_2^2 with its respective equation

In the figure 4.43 we can observe 5 different collisions that produce different results, the equations indicated are:

1. $g_1 \leftrightarrow g_2^2 = g_2$
2. $g_1 \leftrightarrow g_2^2 = g_2$
3. $g_1 \leftrightarrow g_2^2 = g_1$
4. $g_1 \leftrightarrow g_2^2 = g_6 + g_4$
5. $g_1 \leftrightarrow g_2^2 = g_2^2$

Gliders: g_1 and $g_6 + g_4$

In the figure 4.44 we can observe the collisions obtained with the gliders g_1 and $g_6 + g_4$ respectively, below each image is the equation that describes them.

In the figure 4.44 we can observe 5 different collisions that produce different results, the equations indicated are:

1. $g_1 \leftrightarrow g_6 + g_4 = g_6 g_4$
2. $g_1 \leftrightarrow g_6 + g_4 = g_1$
3. $g_1 \leftrightarrow g_6 + g_4 = g_6 + g_4$
4. $g_1 \leftrightarrow g_6 + g_4 = gun$
5. $g_1 \leftrightarrow g_6 + g_4 = g_6 + g_4$

Gliders: g_1 and g_6^2

In the figure 4.45 we can observe the collisions obtained with the gliders g_1 y g_6^2 respectively, below each image is the equation that describes them.

In the figure 4.45 we can observe 5 different collisions that produce different results, the equations indicated are:

1. $g_1 \leftrightarrow g_6^2 = g_6 + s_2$

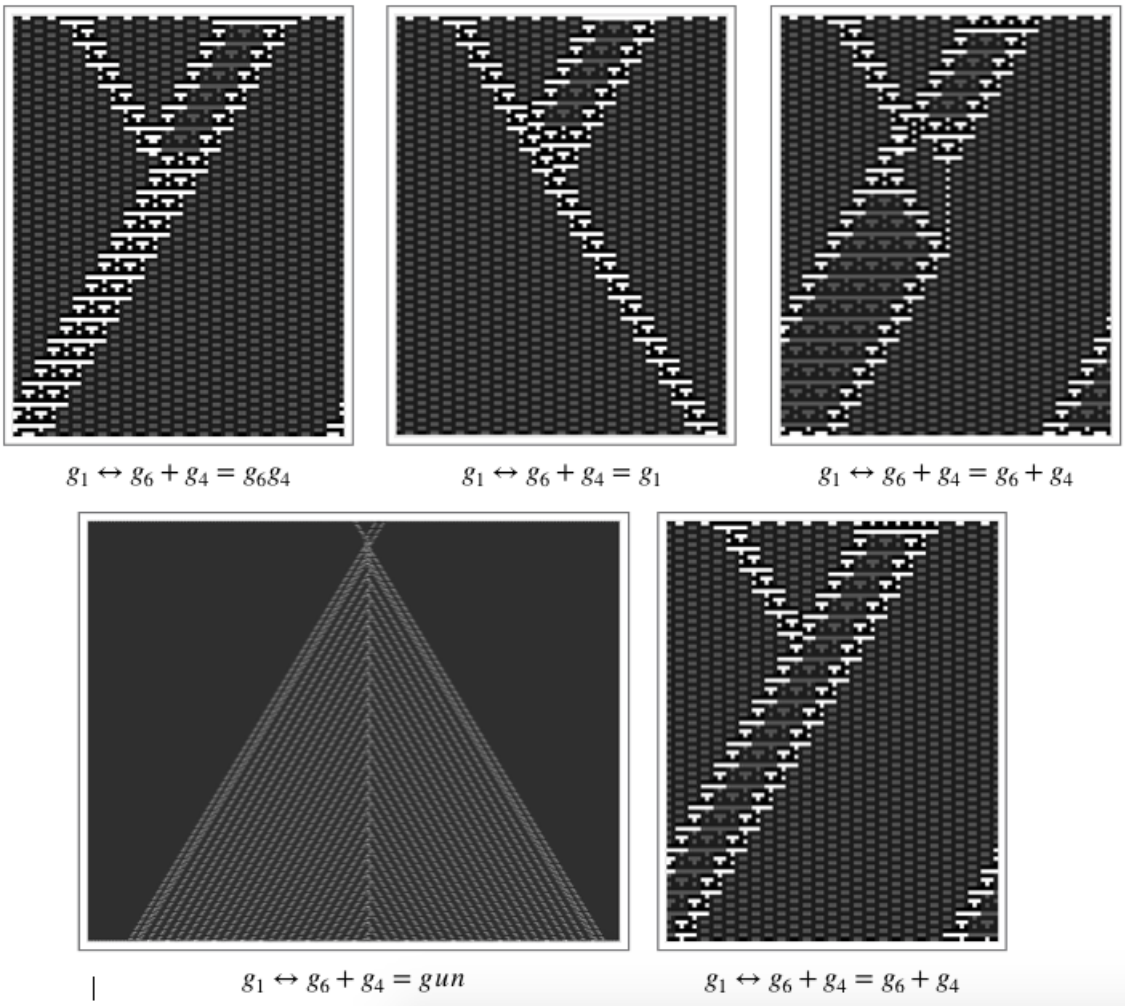


Figura 4.44: The different results obtained from colliding g_1 and $g_6 + g_4$ with its respective equation

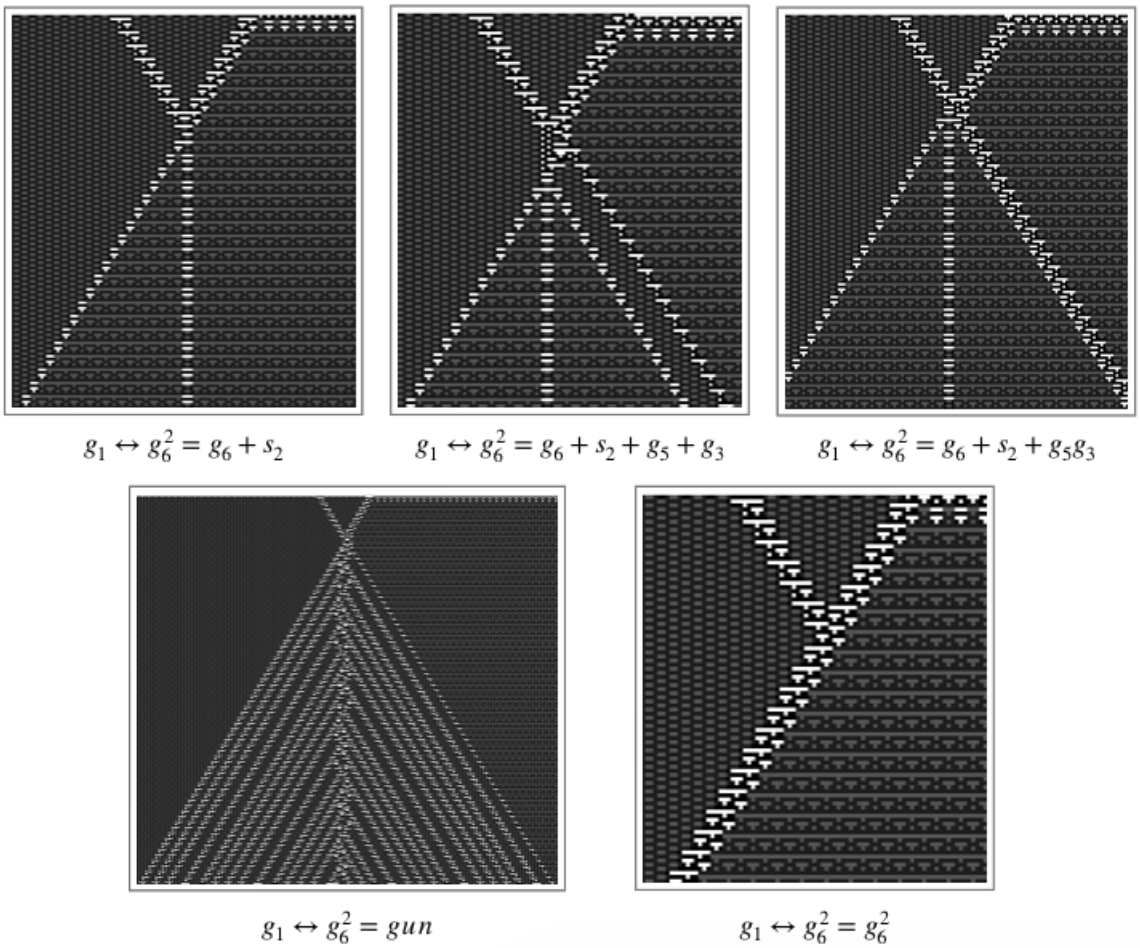


Figura 4.45: The different results obtained from colliding g_1 and g_6^2 with its respective equation

2. $g_1 \leftrightarrow g_6^2 = g_6 + s_2 + g_5 + g_3$
3. $g_1 \leftrightarrow g_6^2 = g_6 + s_2 + g_5 g_3$
4. $g_1 \leftrightarrow g_6^2 = gun$
5. $g_1 \leftrightarrow g_6^2 = g_6^2$

Gliders: g_1 and $g_6 g_4$

In the figure 4.46 we can observe the collisions obtained with the gliders g_1 and $g_6 g_4$ respectively, below each image is the equation that describes them.

In the figure 4.46 we can observe 10 different collisions that produce different results, the equations indicated are:

1. $g_1 \leftrightarrow g_6 g_4 = g_2$
2. $g_1 \leftrightarrow g_6 g_4 = g_6 + g_5$
3. $g_1 \leftrightarrow g_6 g_4 = g_6 g_4$
4. $g_1 \leftrightarrow g_6 g_4 = g_1$
5. $g_1 \leftrightarrow g_6 g_4 = g_6 g_4$
6. $g_1 \leftrightarrow g_6 g_4 = g_2 + g_6 + g_5 + g_1$
7. $g_1 \leftrightarrow g_6 g_4 = g_6 + g_5$
8. $g_1 \leftrightarrow g_6 g_4 = g_6 + g_4$
9. $g_1 \leftrightarrow g_6 g_4 = gun$
10. $g_1 \leftrightarrow g_6 g_4 = gun$

Gliders: g_1, s_1 and g_2

In the figure 4.47 we can observe the collisions obtained using g_1, s_1 and g_2 respectively, below each image is the equation that describes them.

In the figure 4.47 we can only observe a different collision, which equation is:

1. $g_1 \leftrightarrow s_1 \leftrightarrow g_2 = s_1$

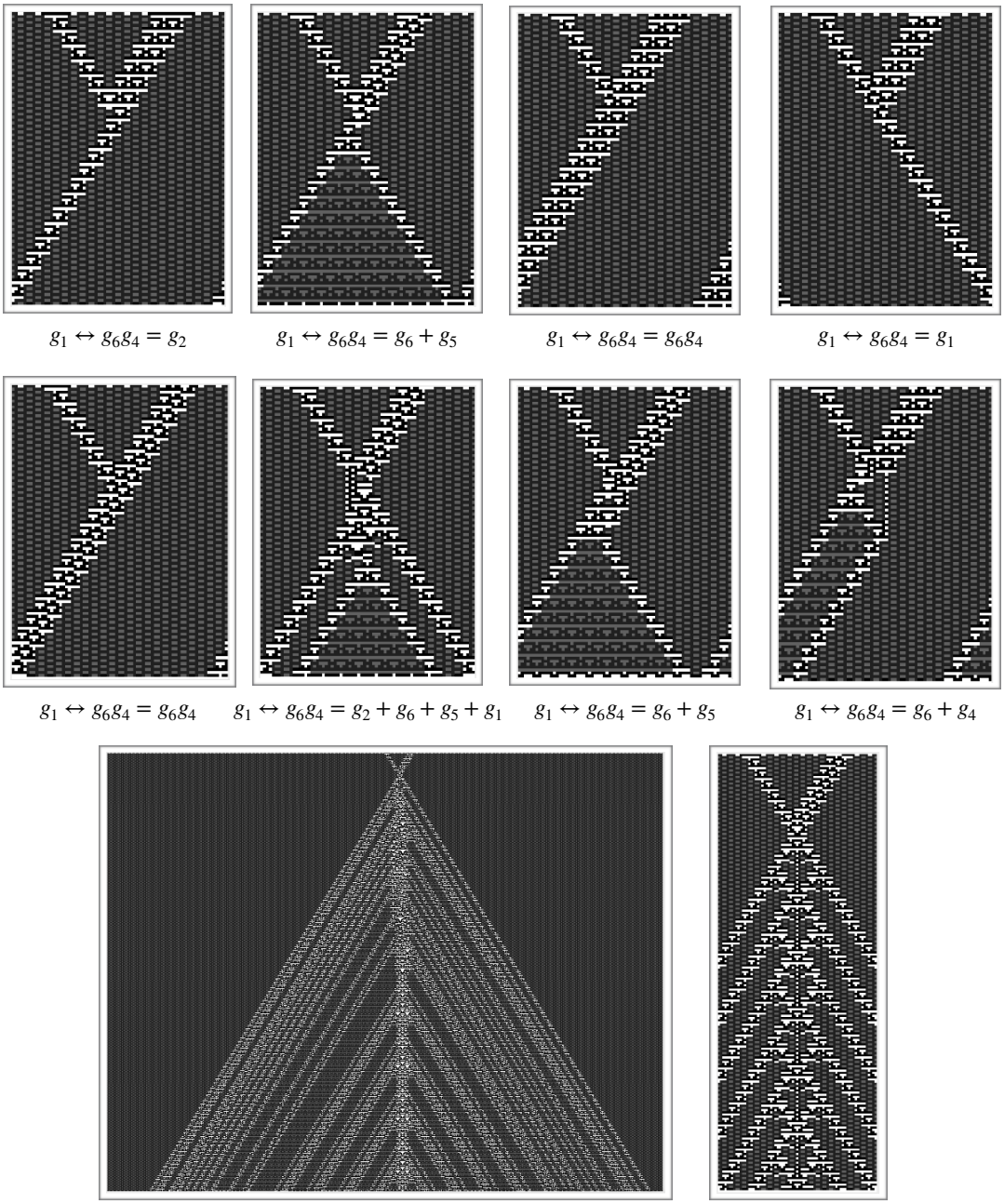
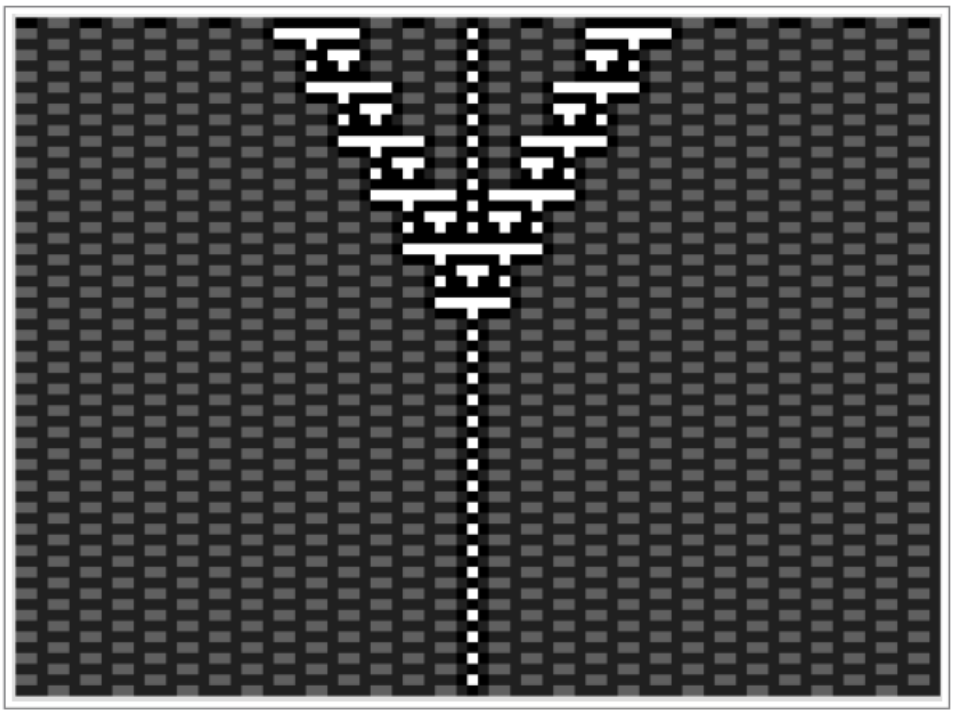


Figura 4.46: The different results obtained from colliding g_1 and g_6g_4 with its respective equation



$$g_1 \leftrightarrow s_1 \leftrightarrow g_2 = s_1$$

Figura 4.47: The different results obtained from colliding g_1, s_1 and g_2 with its respective equation

Gliders: g_1, s_1 and g_6

In the gure 4.48 we can observe the collisions obtained using g_1, s_1 and g_6 respectively, below each image is the equation that describes them.

Figura 4.48: The di erent results obtained from colliding g_1, s_1 and g_6 with its respective equation

In the gure 4.48 we can observe two di erent collisions, whose equations are:

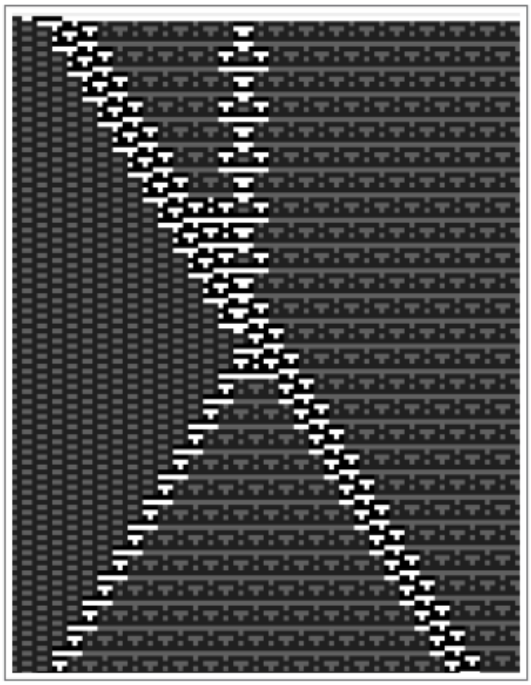
1. $g_1 \$ s_1 \$ g_6 = g_3$
2. $g_1 \$ s_1 \$ g_6 = g_1 + g_3$

Gliders: g_1^2 and g_2

In the gure 4.49 we can observe the collisions obtained using g_1^2 and g_2 respectively, below each image is the equation that describes them.

In the gure 4.49 we can observe 5 di erent collisions that produce di erent results, the equations indicated are:

1. $g_1^2 \$ g_2 = g_2$
2. $g_1^2 \$ g_2 = g_1$



$g_3^2 \leftrightarrow s_2 = g_6 + g_5 g_3$

Figura 4.61: The different results obtained from colliding g_3^2 and s_2 with its respective equation

Gliders: g_5 and $2g_2$

In the figure 4.62 we can observe the collisions obtained using g_5 and $2g_2$ respectively, below each image is the equation that describes them.

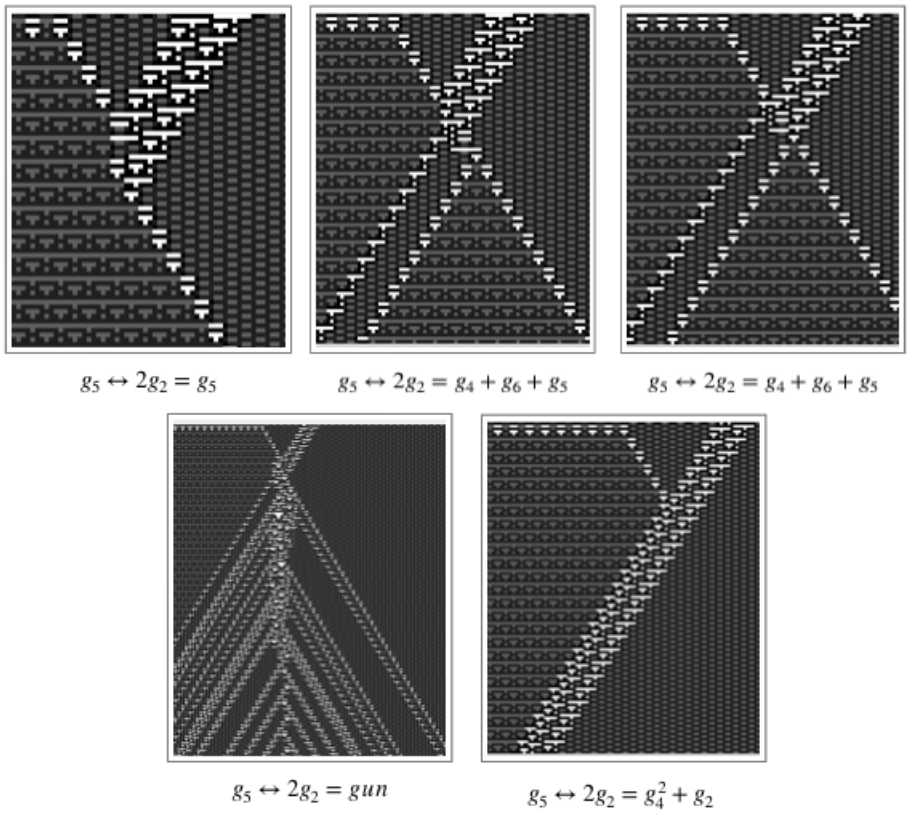


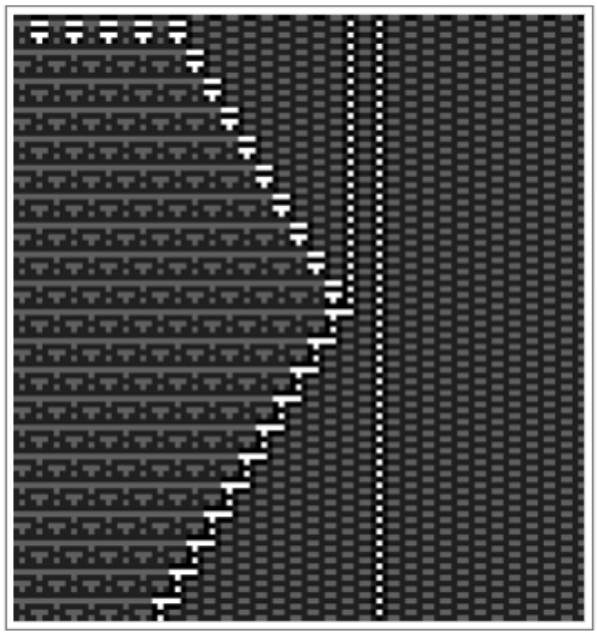
Figura 4.62: The different results obtained from colliding g_5 and $2g_2$ with its respective equation

In the figure 4.62 we can observe 5 different equations, which are:

1. $g_5 \leftrightarrow 2g_2 = g_5$
2. $g_5 \leftrightarrow 2g_2 = g_4 + g_6 + g_5$
3. $g_5 \leftrightarrow 2g_2 = g_4 + g_6 + g_5$
4. $g_5 \leftrightarrow 2g_2 = gun$
5. $g_5 \leftrightarrow 2g_2 = g_4^2 + g_2$

Gliders: g_5 and $2s_1$

In the figure 4.63 we can observe the collisions obtained using g_5 and $2s_1$ respectively, below each image is the equation that describes them.



$g_5 \leftrightarrow 2s_1 = s_1$

Figura 4.63: The different results obtained from colliding g_5 and $2s_1$ with its respective equation

In the figure 4.63 we can observe a different equation, which is:

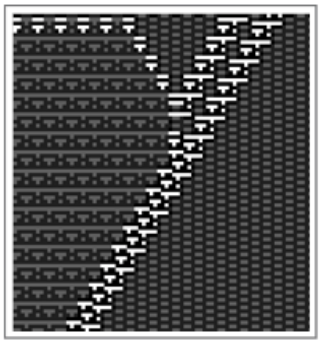
1. $g_5 \leftrightarrow 2s_1 = s_1$

Gliders: g_5, g_2 and g_2

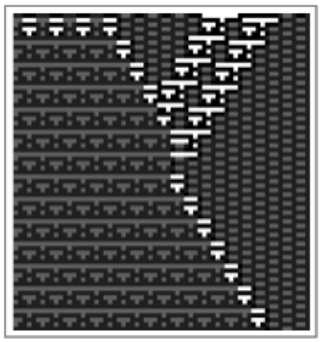
In the figure 4.64 we can observe the collisions obtained using g_5 and $g_2 + g_2$ respectively, below each image is the equation that describes them.

In the figure 4.64 we can observe 5 different equations, which are:

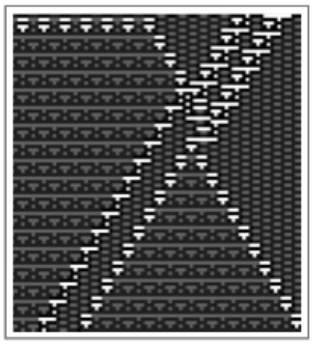
1. $g_5 \leftrightarrow g_2 + g_2 = g_4^2$



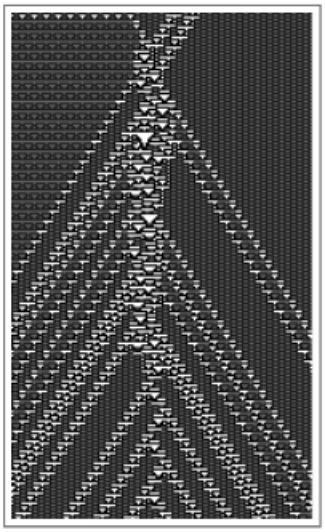
$g_5 \leftrightarrow g_2 + g_2 = g_4^2$



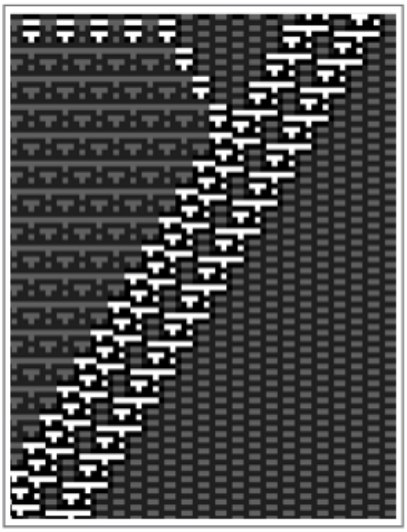
$g_5 \leftrightarrow g_2 + g_2 = g_5$



$g_5 \leftrightarrow g_2 + g_2 = g_4 + g_6 + g_5$



$g_5 \leftrightarrow g_2 + g_2 = gun$



$g_5 \leftrightarrow g_2 + g_2 = g_4^2 + g_2$

Figura 4.64: The different results obtained from colliding g_5 and $g_2 + g_2$ with its respective equation

2. $g_5 \leftrightarrow g_2 + g_2 = g_5$
3. $g_5 \leftrightarrow g_2 + g_2 = g_4 + g_6 + g_5$
4. $g_5 \leftrightarrow g_2 + g_2 = gun$
5. $g_5 \leftrightarrow g_2 + g_2 = g_4^2 + g_2$

Gliders: g_5 and g_2^2

In the figure 4.65 we can observe the collisions obtained using g_5 and g_2^2 respectively, below each image is the equation that describes them.

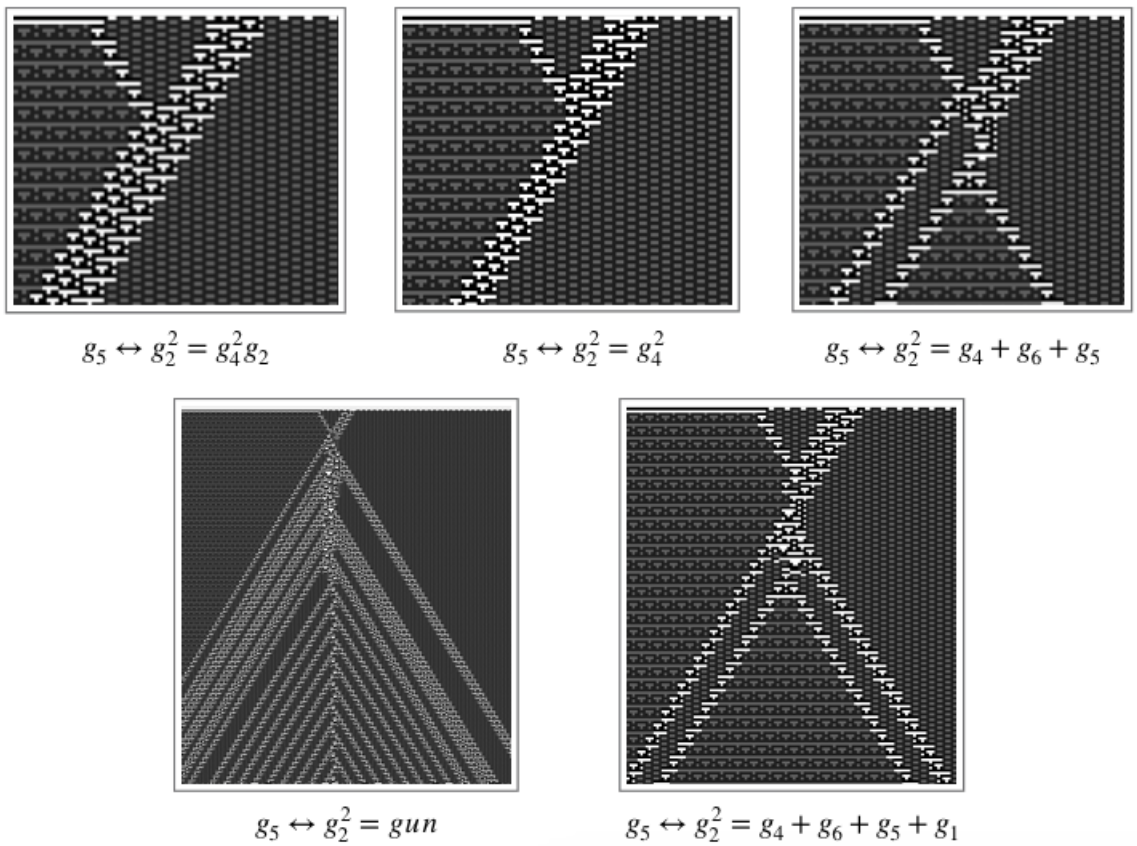


Figura 4.65: The different results obtained from colliding g_5 and g_2^2 with its respective equation

In the figure 4.65 we can observe 5 different equations, which are:

1. $g_5 \leftrightarrow g_2^2 = g_4^2 g_2$
2. $g_5 \leftrightarrow g_2^2 = g_4^2$
3. $g_5 \leftrightarrow g_2^2 = g_4 + g_6 + g_5$
4. $g_5 \leftrightarrow g_2^2 = gun$
5. $g_5 \leftrightarrow g_2^2 = g_4 + g_6 + g_5 + g_1$

Gliders: g_5 and $g_6 + g_4$

In the figure 4.66 we can observe the collisions obtained using g_5 and $g_6 + g_4$ respectively, below each image is the equation that describes them.

In the figure 4.66 we can observe 5 different equations, which are:

1. $g_5 \leftrightarrow g_6 + g_4 = g_5$
2. $g_5 \leftrightarrow g_6 + g_4 = g_5$
3. $g_5 \leftrightarrow g_6 + g_4 = g_4 g_6 + g_4$
4. $g_5 \leftrightarrow g_6 + g_4 = chaos$
5. $g_5 \leftrightarrow g_6 + g_4 = gun$

Gliders: g_5 and g_6^2

In the figure 4.67 we can observe the collisions obtained using g_5 and g_6^2 respectively, below each image is the equation that describes them.

In the figure 4.67 we can observe 5 different equations, which are:

1. $g_5 \leftrightarrow g_6^2 = g_4 g_6 + s_2$
2. $g_5 \leftrightarrow g_6^2 = g_4 + g_6 + 2s_2 + g_5 + g_1 g_3$
3. $g_5 \leftrightarrow g_6^2 = g_4 g_6^2$
4. $g_5 \leftrightarrow g_6^2 = g_4 + g_6 + 2s_2 + g_5 + g_3$
5. $g_5 \leftrightarrow g_6^2 = g_4 + g_2 + g_6 + 2s_2 + g_5 + g_1 + g_3$

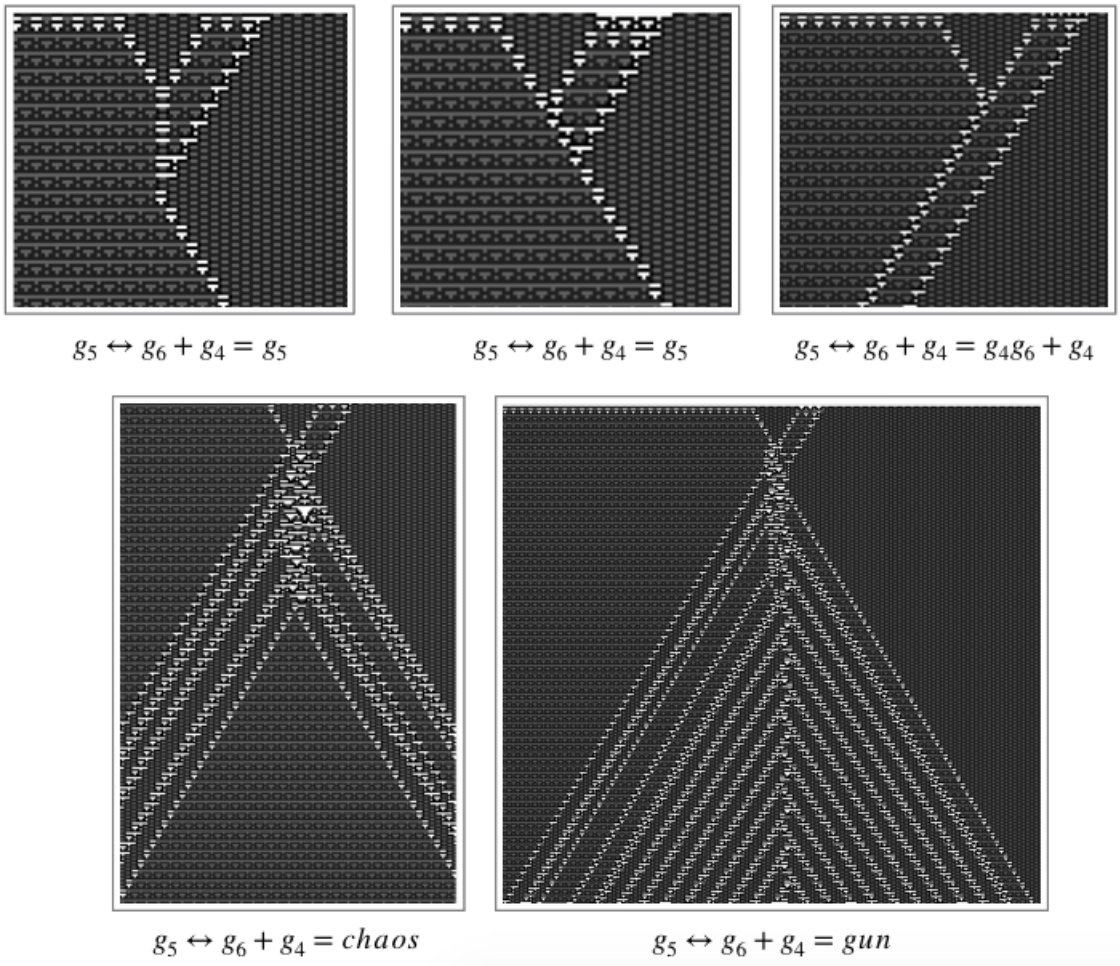


Figura 4.66: The different results obtained from colliding g_5 and $g_6 + g_4$ with its respective equation

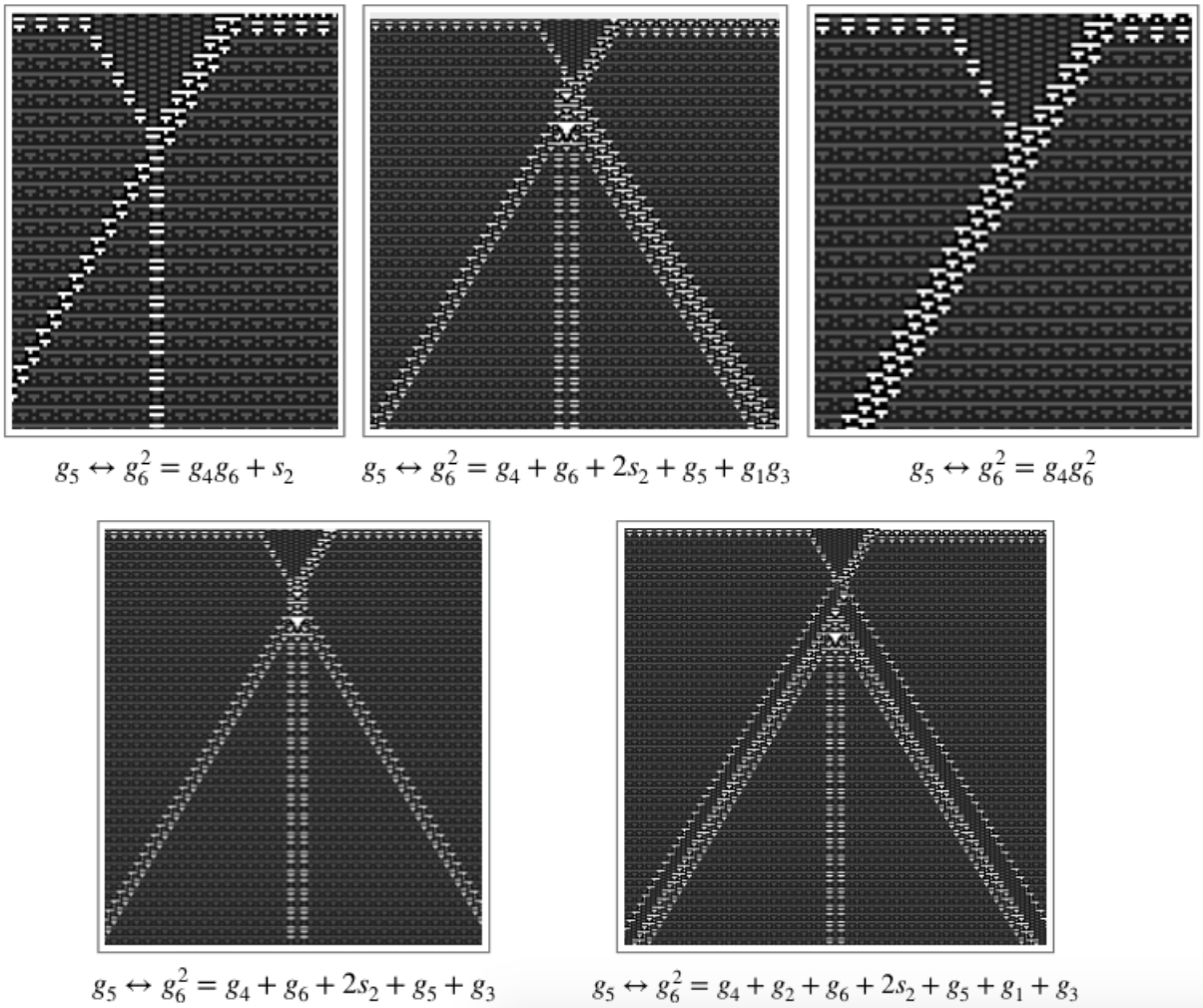


Figura 4.67: The different results obtained from colliding g_5 and g_6^2 with its respective equation

Gliders: g_5 and g_6g_4

In the figure 4.68 we can observe the collisions obtained using g_5 and g_6g_4 respectively, below each image is the equation that describes them.

In the figure 4.68 we can observe 10 different equations, which are:

1. $g_5 \leftrightarrow g_6g_4 = gun$
2. $g_5 \leftrightarrow g_6g_4 = gun$
3. $g_5 \leftrightarrow g_6g_4 = g_4^3$
4. $g_5 \leftrightarrow g_6g_4 = g_6g_4$
5. $g_5 \leftrightarrow g_6g_4 = g_4g_6 + g_5$
6. $g_5 \leftrightarrow g_6g_4 = g_5$
7. $g_5 \leftrightarrow g_6g_4 = g_5$
8. $g_5 \leftrightarrow g_6g_4 = gun$
9. $g_5 \leftrightarrow g_6g_4 = gun$
10. $g_5 \leftrightarrow g_6g_4 = gun$

Gliders: g_5, s_1 and g_2

In the figure 4.69 we can observe the collisions obtained using g_5, s_1 and g_2 respectively, below each image is the equation that describes them.

In the figure 4.69 we can observe 2 different equations, which are:

1. $g_5 \leftrightarrow s_1 \leftrightarrow g_2 = g_4$
2. $g_5 \leftrightarrow s_1 \leftrightarrow g_2 = g_4g_2$

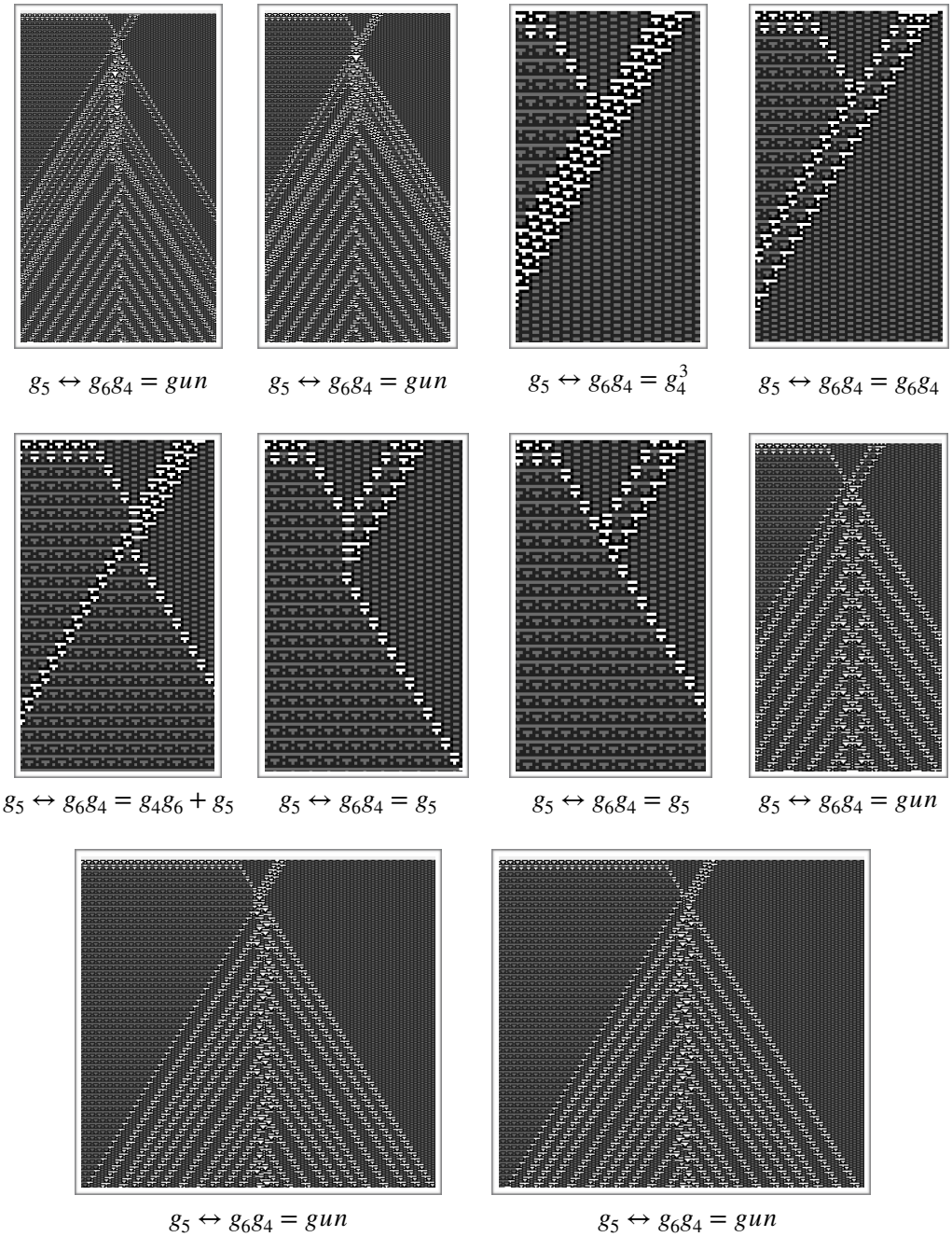
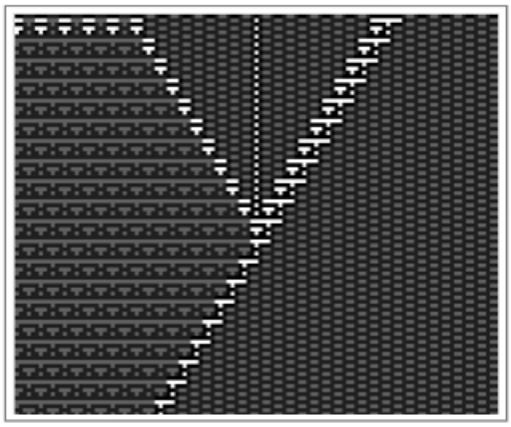
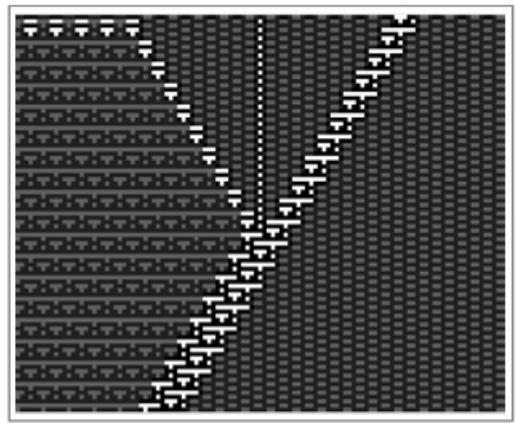


Figura 4.68: The different results obtained from colliding g_5 and g_6g_4 with its respective equation

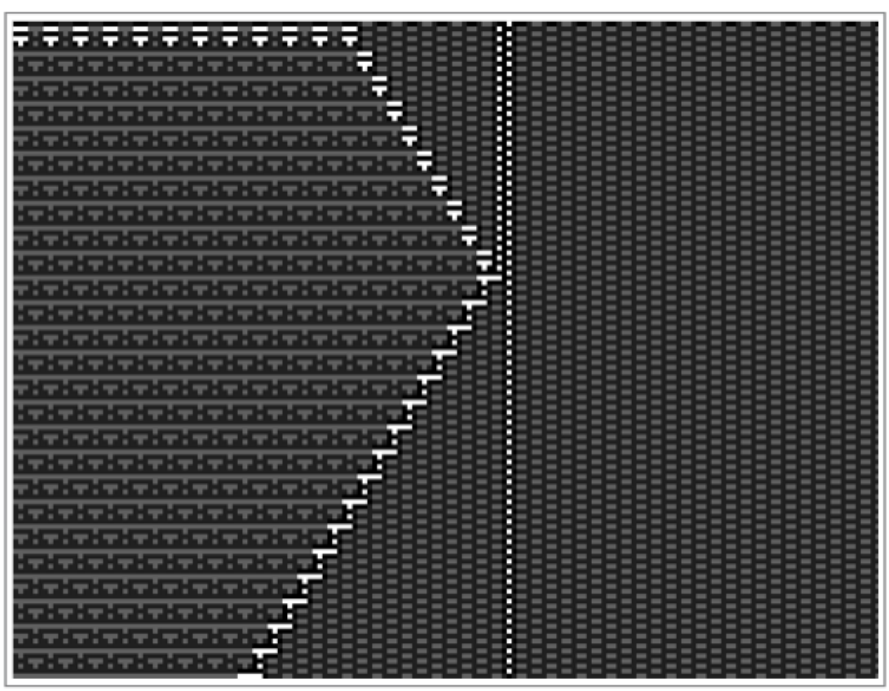


$g_5 \leftrightarrow s_1 \leftrightarrow g_2 = g_4$



$g_5 \leftrightarrow s_1 \leftrightarrow g_2 = g_4 g_2$

Figura 4.69: The different results obtained from colliding g_5, s_1 and g_2 with its respective equation



$g_5 \leftrightarrow s_1^2 = g_4 + s_1$

Figura 4.70: The different results obtained from colliding g_5 and s_1^2 with its respective equation

Gliders: g_5 and s_1^2

In the figure 4.70 we can observe the collisions obtained using g_5 and s_1^2 respectively, below each image is the equation that describes them.

In the figure 4.70 we can observe a different equation, which is:

1. $g_5 \leftrightarrow s_1^2 = g_4 + s_1$

Gliders: g_5^2 and g_2

In the figure 4.71 we can observe the collisions obtained using g_5^2 and g_2 respectively, below each image is the equation that describes them.

In the figure 4.71 we can observe 5 different equations, which are:

1. $g_5^2 \leftrightarrow g_2 = s_2 + g_5$
2. $g_5^2 \leftrightarrow g_2 = g_5^2$
3. $g_5^2 \leftrightarrow g_2 = g_4 g_6 + s_2 + g_5$
4. $g_5^2 \leftrightarrow g_2 = g_4 + g_6 + s_2 + g_5$
5. $g_5^2 \leftrightarrow g_2 = gun$

Gliders: g_5^2 and g_6

In the figure 4.72 we can observe the collisions obtained using g_5^2 and g_6 respectively, below each image is the equation that describes them.

In the figure 4.72 we can observe 5 different equations, which are:

1. $g_5^2 \leftrightarrow g_6 = g_5 g_3$
2. $g_5^2 \leftrightarrow g_6 = g_5^2 g_3$
3. $g_5^2 \leftrightarrow g_6 = g_4 g_2 + g_6 + 2s_2 + g_5 + g_3$
4. $g_5^2 \leftrightarrow g_6 = g_4 + g_2 + g_6 + 2s_2 + g_5 + g_1 + g_3$
5. $g_5^2 \leftrightarrow g_6 = g_4 + g_6 + 2s_2 + g_5 + g_3$

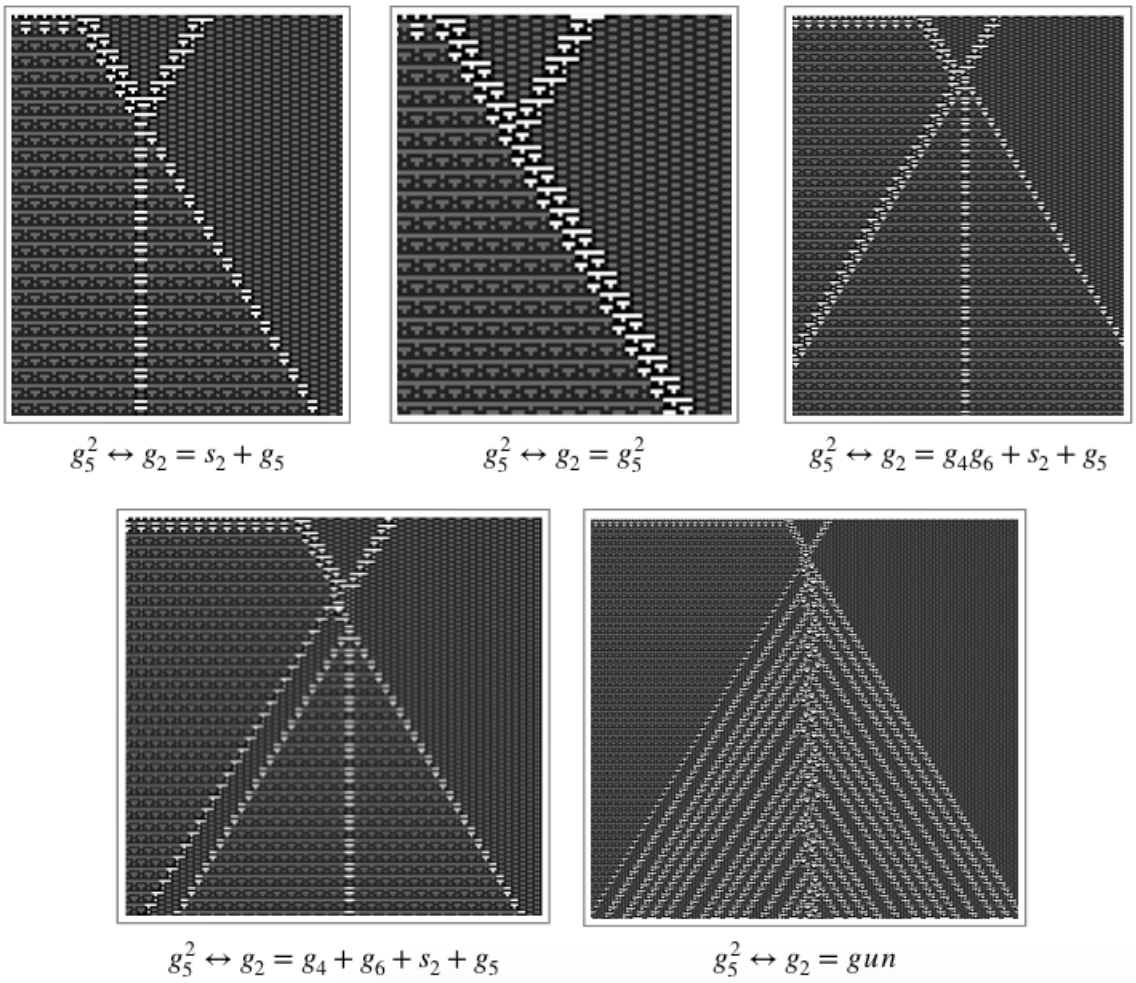
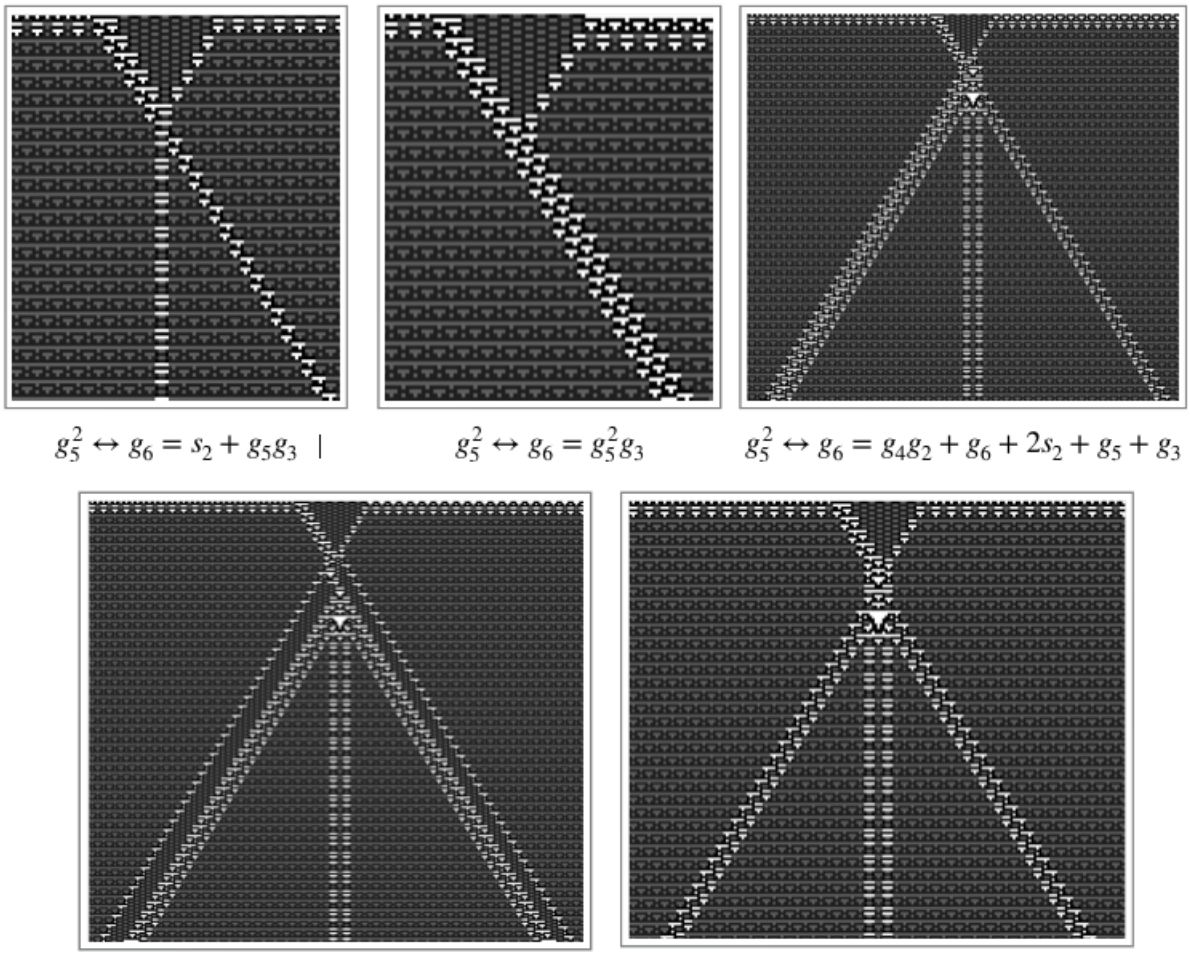


Figura 4.71: The different results obtained from colliding g_5^2 and g_2 with its respective equation



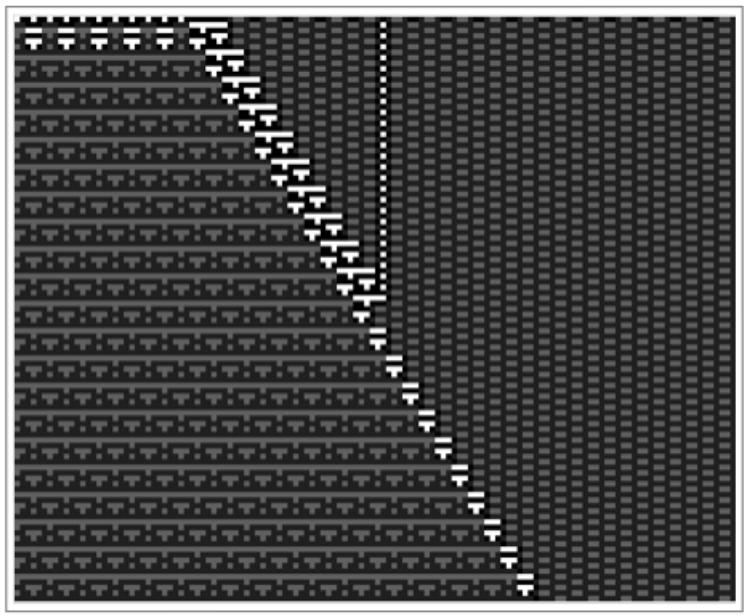
$g_5^2 \leftrightarrow g_6 = s_2 + g_5 g_3$ | $g_5^2 \leftrightarrow g_6 = g_5^2 g_3$ $g_5^2 \leftrightarrow g_6 = g_4 g_2 + g_6 + 2s_2 + g_5 + g_3$

$g_5^2 \leftrightarrow g_6 = g_4 + g_2 + g_6 + 2s_2 + g_5 + g_1 + g_3$ $g_5^2 \leftrightarrow g_6 = g_4 + g_6 + 2s_2 + g_5 + g_3$

Figura 4.72: The different results obtained from colliding g_5^2 and g_6 with its respective equation

Gliders: g_5^2 and s_1

In the figure 4.73 we can observe the collisions obtained using g_5^2 and s_1 respectively, below each image is the equation that describes them.



$g_5^2 \leftrightarrow s_1 = g_5$

Figura 4.73: The different results obtained from colliding g_5^2 and s_1 with its respective equation

In the figure 4.73 we can observe a different equation, which is:

1. $g_5^2 \leftrightarrow s_1 = g_5$

Summary of ternary collisions

In the table 4.12 all the equations of the collisions that we exemplify are summarized. In total there are 915 ternary collisions carried out, this was calculated using what was obtained in the binary collisions and was reduced to a case study (more details in the appendix B).

Equation	**Result**
$2g_1 \leftrightarrow g_2$	$2g_1$
$2g_1 \leftrightarrow g_2$	$g_3 + g_5$
$2g_1 \leftrightarrow g_2$	$g_6 + g_4$
$2g_1 \leftrightarrow g_2$	$g_6 + g_4$
$2g_1 \leftrightarrow g_2$	g_1
$2g_1 \leftrightarrow g_6$	$g_1 + g_3^2$
$2g_1 \leftrightarrow g_6$	gun
$2g_1 \leftrightarrow g_6$	$g_6 + g_5 + g_3$
$2g_1 \leftrightarrow g_6$	$g_6 + g_5 + g_3$
$2g_1 \leftrightarrow g_6$	g_6
$2g_1 \leftrightarrow s_1$	s_1
$g_1 + g_1 \leftrightarrow g_2$	$\emptyset$
$g_1 + g_1 \leftrightarrow g_2$	g_1
$g_1 + g_1 \leftrightarrow g_2$	$g_1 + g_1$
$g_1 + g_1 \leftrightarrow g_2$	$g_3 + g_5$
$g_1 + g_1 \leftrightarrow g_2$	g_2
$g_1 + g_1 \leftrightarrow g_6$	g_6
$g_1 + g_1 \leftrightarrow g_6$	g_3^2
$g_1 + g_1 \leftrightarrow g_6$	$g_1 + g_3^2$
$g_1 + g_1 \leftrightarrow g_6$	gun
$g_1 + g_1 \leftrightarrow g_6$	$g_6 + g_5 + g_3$
$g_1 + g_1 \leftrightarrow s_1$	s_1
$g_1 \leftrightarrow 2g_2$	g_2
$g_1 \leftrightarrow 2g_2$	$g_3 + g_5$
$g_1 \leftrightarrow 2g_2$	$g_3 + g_5$
$g_1 \leftrightarrow 2g_2$	$g_6 + g_4$
$g_1 \leftrightarrow 2g_2$	$2g_2$
$g_1 \leftrightarrow 2s_1$	$s_1 + s_1$
$g_1 \leftrightarrow g_2 + g_2$	g_2
$g_1 \leftrightarrow g_2 + g_2$	$\emptyset$

Continued on the next page

Cuadro 4.12 – *Continued from previous page*

Equation	**Result**
$g_1 \leftrightarrow g_2 + g_2$	g_1
$g_1 \leftrightarrow g_2 + g_2$	$g_6 + g_4$
$g_1 \leftrightarrow g_2 + g_2$	$g_2 + g_2$
$g_1 \leftrightarrow g_2^2$	g_2
$g_1 \leftrightarrow g_2^2$	g_2
$g_1 \leftrightarrow g_2^2$	g_1
$g_1 \leftrightarrow g_2^2$	$g_6 + g_4$
$g_1 \leftrightarrow g_2^2$	g_2^2
$g_1 \leftrightarrow g_6 + g_4$	$g_6 g_4$
$g_1 \leftrightarrow g_6 + g_4$	g_1
$g_1 \leftrightarrow g_6 + g_4$	$g_6 + g_4$
$g_1 \leftrightarrow g_6 + g_4$	gun
$g_1 \leftrightarrow g_6 + g_4$	$g_6 + g_4$
$g_1 \leftrightarrow g_6^2$	$g_6 + s_2$
$g_1 \leftrightarrow g_6^2$	$g_6 + s_2 + g_5 + g_3$
$g_1 \leftrightarrow g_6^2$	$g_6 + s_2 + g_5 g_3$
$g_1 \leftrightarrow g_6^2$	gun
$g_1 \leftrightarrow g_6^2$	g_6^2
$g_1 \leftrightarrow g_6 g_4$	g_2
$g_1 \leftrightarrow g_6 g_4$	$g_6 + g_5$
$g_1 \leftrightarrow g_6 g_4$	$g_6 g_4$
$g_1 \leftrightarrow g_6 g_4$	g_1
$g_1 \leftrightarrow g_6 g_4$	$g_6 g_4$
$g_1 \leftrightarrow g_6 g_4$	$g_2 + g_6 + g_5 + g_1$
$g_1 \leftrightarrow g_6 g_4$	$g_6 + g_5$
$g_1 \leftrightarrow g_6 g_4$	$g_6 + g_4$
$g_1 \leftrightarrow g_6 g_4$	gun
$g_1 \leftrightarrow g_6 g_4$	gun
$g_1 \leftrightarrow s_1 \leftrightarrow g_2$	s_1

Continued on the next page

Cuadro 4.12 – *Continued from previous page*

Equation	**Result**
$g_1 \leftrightarrow s_1 \leftrightarrow g_6$	g_3
$g_1 \leftrightarrow s_1 \leftrightarrow g_6$	$g_1 + g_3$
$g_1^2 \leftrightarrow g_2$	g_2
$g_1^2 \leftrightarrow g_2$	g_1
$g_1^2 \leftrightarrow g_2$	g_1
$g_1^2 \leftrightarrow g_2$	g_1^2
$g_1^2 \leftrightarrow g_2$	$g_3 + g_5$
$g_1^2 \leftrightarrow g_6$	$g_6 + g_5 + g_3$
$g_1^2 \leftrightarrow g_6$	g_3^2
$g_1^2 \leftrightarrow g_6$	$g_2 + g_6 + g_5 + g_3$
$g_1^2 \leftrightarrow g_6$	$g_1 g_3^2$
$g_1^2 \leftrightarrow g_6$	gun
$g_1^2 \leftrightarrow s_1$	s_1
$g_3 \leftrightarrow 2s_2$	$g_6 + s_2$
$g_3 \leftrightarrow g_4 + g_2$	s_1
$g_3 \leftrightarrow g_4 + g_6$	g_3
$g_3 \leftrightarrow g_4^2$	g_2
$g_3 \leftrightarrow g_4 g_2$	s_1
$g_3 \leftrightarrow s_2 + s_2$	$g_6 + s_2$
$g_3 \leftrightarrow s_2 \leftrightarrow g_4$	$g_6 + g_4$
$g_3 \leftrightarrow s_2 \leftrightarrow g_4$	g_2
$g_3 \leftrightarrow s_2 \leftrightarrow g_4$	$\emptyset$
$g_3 \leftrightarrow s_2 \leftrightarrow g_4$	g_1
$g_3 \leftrightarrow s_2 \leftrightarrow g_4$	$g_4 + g_5$
$g_3 \leftrightarrow s_2^2$	$g_6 + s_2$
$g_3^2 \leftrightarrow g_4$	g_1
$g_3^2 \leftrightarrow s_2$	$g_6 + g_5 g_3$
$g_5 \leftrightarrow 2g_2$	g_5
$g_5 \leftrightarrow 2g_2$	$g_4 + g_6 + g_5$

Continued on the next page

Cuadro 4.12 – *Continued from previous page*

Equation	**Result**
$g_5 \leftrightarrow 2g_2$	$g_4 + g_6 + g_5$
$g_5 \leftrightarrow 2g_2$	gun
$g_5 \leftrightarrow 2g_2$	$g_4^2 + g_2$
$g_5 \leftrightarrow 2s_1$	s_1
$g_5 \leftrightarrow g_2 + g_2$	g_4^2
$g_5 \leftrightarrow g_2 + g_2$	g_5
$g_5 \leftrightarrow g_2 + g_2$	$g_4 + g_6 + g_5$
$g_5 \leftrightarrow g_2 + g_2$	gun
$g_5 \leftrightarrow g_2 + g_2$	$g_4^2 + g_2$
$g_5 \leftrightarrow g_2^2$	$g_4^2 g_2$
$g_5 \leftrightarrow g_2^2$	g_4^2
$g_5 \leftrightarrow g_2^2$	$g_4 + g_6 + g_5$
$g_5 \leftrightarrow g_2^2$	gun
$g_5 \leftrightarrow g_2^2$	$g_4 + g_6 + g_5 + g_1$
$g_5 \leftrightarrow g_6 + g_4$	g_5
$g_5 \leftrightarrow g_6 + g_4$	g_5
$g_5 \leftrightarrow g_6 + g_4$	$g_4 g_6 + g_4$
$g_5 \leftrightarrow g_6 + g_4$	$chaos$
$g_5 \leftrightarrow g_6 + g_4$	gun
$g_5 \leftrightarrow g_6^2$	$g_4 g_6 + s_2$
$g_5 \leftrightarrow g_6^2$	$g_4 + g_6 + 2s_2 + g_5 + g_1 g_3$
$g_5 \leftrightarrow g_6^2$	$g_4 g_6^2$
$g_5 \leftrightarrow g_6^2$	$g_4 + g_6 + 2s_2 + g_5 + g_3$
$g_5 \leftrightarrow g_6^2$	$g_4 + g_2 + g_6 + 2s_2 + g_5 + g_1 + g_3$
$g_5 \leftrightarrow g_6 g_4$	gun
$g_5 \leftrightarrow g_6 g_4$	gun
$g_5 \leftrightarrow g_6 g_4$	g_4^3
$g_5 \leftrightarrow g_6 g_4$	$g_6 g_4$
$g_5 \leftrightarrow g_6 g_4$	$g_4 g_6 + g_5$

Continued on the next page

Cuadro 4.12 – *Continued from previous page*

Equation	**Result**
$g_5 \leftrightarrow g_6 g_4$	g_5
$g_5 \leftrightarrow g_6 g_4$	g_5
$g_5 \leftrightarrow g_6 g_4$	gun
$g_5 \leftrightarrow g_6 g_4$	gun
$g_5 \leftrightarrow g_6 g_4$	gun
$g_5 \leftrightarrow s_1 \leftrightarrow g_2$	g_4
$g_5 \leftrightarrow s_1 \leftrightarrow g_2$	$g_4 g_2$
$g_5 \leftrightarrow s_1^2$	$g_4 + s_1$
$g_5^2 \leftrightarrow g_2$	$s_2 + g_5$
$g_5^2 \leftrightarrow g_2$	g_5^2
$g_5^2 \leftrightarrow g_2$	$g_4 g_6 + s_2 + g_5$
$g_5^2 \leftrightarrow g_2$	$g_4 + g_6 + s_2 + g_5$
$g_5^2 \leftrightarrow g_2$	gun
$g_5^2 \leftrightarrow g_6$	$g_5 g_3$
$g_5^2 \leftrightarrow g_6$	$g_5^2 g_3$
$g_5^2 \leftrightarrow g_6$	$g_4 g_2 + g_6 + 2s_2 + g_5 + g_3$
$g_5^2 \leftrightarrow g_6$	$g_4 + g_2 + g_6 + 2s_2 + g_5 + g_1 + g_3$
$g_5^2 \leftrightarrow g_6$	$g_4 + g_6 + 2s_2 + g_5 + g_3$
$g_5^2 \leftrightarrow s_1$	g_5

Cuadro 4.12: Summary table of ternary collisions

4.6.3. Colisiones de orden superior

In this section we will illustrate some collisions that were made using 4 and 5 particles within the space of evolutions, considered as quaternary and quinary respectively. These collisions were made to illustrate some interesting collisions, as well as show that more particles, more collisions are required, so we face a problem of infinity and not decidable.

Quaternary collisions

In the figure 4.74 we can observe some interesting quaternary collisions, which mostly generate glider-gun.

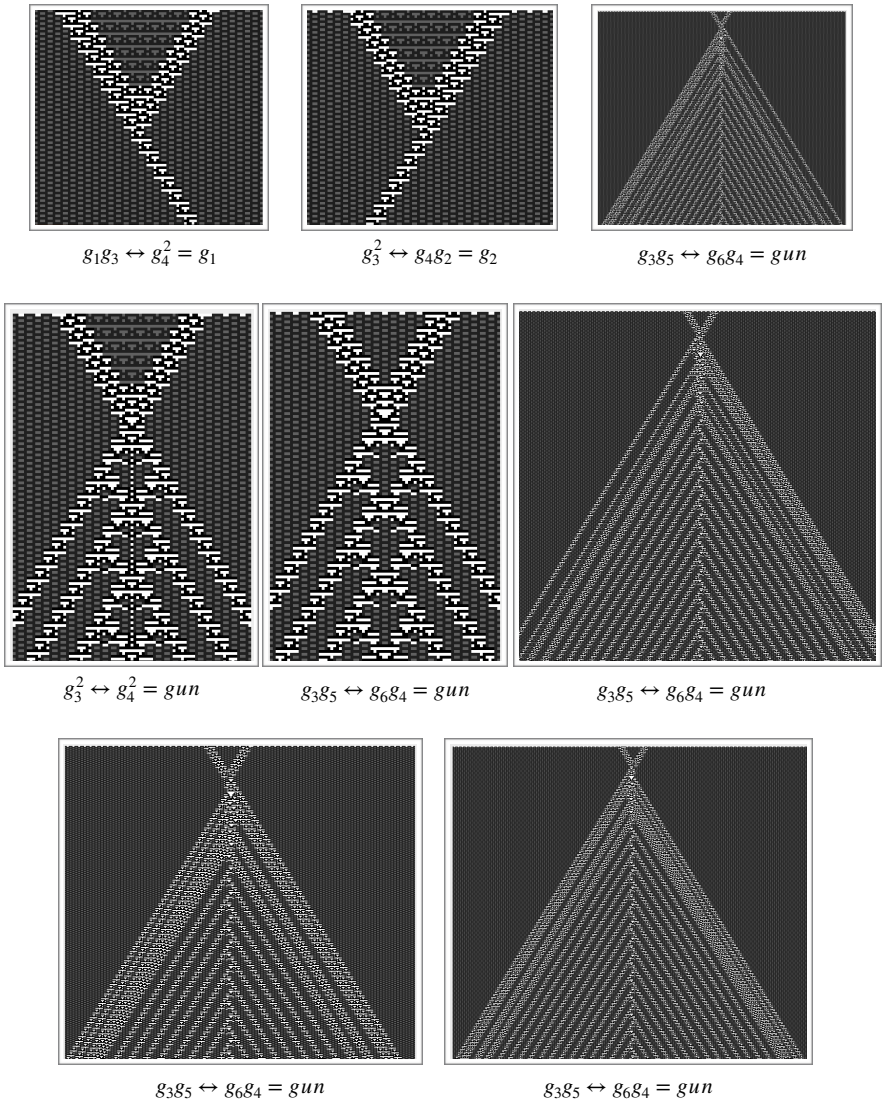


Figura 4.74: Some results of colliding 4 particles within the evolution space

Capítulo 5

Collision-based computing

5.1. Background

To carry out any operation with the computer, programmers, developers, researchers, etc., use the basic units known as bits (0 and 1), which when evaluated within logical gates like AND, OR, XOR, etc, and uniting many of these, we can compute from a sum to a program that allows us to analyze a DNA chain.

Now we will try to do computing in a non-structured environment populated by moving objects, without cables, without valves, there is nothing there: only compact patterns deanbulating in space, colliding with each other and calculating.

A computer device should be general purpose, universal, or specialized. A universal processor can be almost everything; specialized - almost nothing. Personal computers are universal, the controllers of a microwave are specialized. One could study two types of universality - logical, or computational, and simulational. An abstract machine as well as a real physical, chemical or bologic system, is called computationally universal if it implements a functionally complete, or universal, set of Boolean functions in its space-time dynamics. The constructions mostly studied within the collision-based computation are computationally universal, because they carry out universal systems of logic gates in hierarchical collisions of moving objects. If a system simulates the behavior of a universal machine, which its universality has been proven, it is called simulationally universal.

A universal processing device can be either structured, heterogeneous, compart-

mentalized and stationary or without structure, homogeneous, without architecture and dynamic. Structured devices have wires to transmit information, and valves to transmit it: devices without structure do not have any of that. The quanta of information are represented by moving objects (either by their presence / absence or particular types, colors) that travel in space. The trajectories of the objects can be seen as the wires. Objects change their trajectories or states when they collide with other objects. Thus, information is transformed and computing can be implemented.

There are quite a few sources for collision-based computation. Studies that deal with signal collisions, journeys along discrete chains, cellular automata of one dimension, lie at the beginning of the field. The ideas of colliding signals, which have existed since the 19th century in physics and physiology, have been placed in the framework of automata recently, about 1965, when articles by A. J. Atrubin, P. Fisher and A. Waksman were published. It is known that Atrubin studied multiplication in one-dimensional cellular automata; Waksman gave a solution of 8 states for the problem of synchronization of a firing squad; and Fisher showed how to generate prime numbers in cellular automata. The approach of Atrubin-Fisher-Waksman allowed the development of unimaginable constructions aimed at increasing the computing potential of networks of homogeneous automata.

In 1971 E. R. Banks showed how to construct simple wires and gates in configurations of a two-dimensional binary state cellular automaton. This was construction based on architecture. Thus, a wire was represented by a particular stationary configuration of cell states; this was rather a simulation of an electrical circuit, or logical, conventional. The design of Banks was removed from its heterogeneity ten years later, when the Game of Life made our world "without wires"

In 1982 Elwyn Berlekamp, John Conway and Richard Gay proved that the Game of Life can imitate computers. They imitated the electric cables with lines on which the gliders travel, and demonstrated how to design a logic gate colliding gliders with one to another. His "ways to win" brought us admirable computer designs that not only look fresh twenty years later, but are still rediscovered again and again by the game of Life enthusiasts around the network. Berlekamp, Conway and Gay used a fade reaction of gliders - two gliders colliding a annihilating - to build a NOT gate. They adopted Gosper's "dining room" to collect garbage and to destroy the glider

currents. They used combinations of glider-guns and dining rooms to implement AND and OR gates, and changed from a stationary pattern (block) to a mobile pattern (glider) when they designed auxiliary information storage.

"There is even the possibility that space-time is granular, composed of discrete units, and that the universe, as Edward Fredkin of MIT and others have suggested, is a cellular automaton run by a huge computer. If yes, what we call a motion could only be a simulated motion. A moving particle at the last micro level can be essentially the same as one of our gliders, apparently moving at the macro level, where currently there is only an alteration of basic cell-space-time states in obedience to the transition rules that have to be to be discovered yet." - Berlekamp-Conway-Gay's book ends. His last words were about Fredkin.

Meanwhile, in 1978 Edward Fredkin and Tommaso Toffoli uploaded a proposal for a one-year project to DARPA, which was funded, and thus began deploying a chain of remarkable events. Originally, Fredkin and Toffoli directed "drastically reduce the fraction of energy that is dissipated in each computational step." To design a computer without dissipation, they constructed a new type of logical logic - conservative logic - that preserves both "the physical quantities in which the digital signals are encoded " and "the information present at any time in a digital system" . They further developed those ideas in the seminal article Çonservative Logic"published in 1982. Thus, the concept of ballistic computers emerged. The Fredkin-Toffoli model of conservative computation - the billiard ball model - explores "elastic collisions involving balls and fixed reflexors". Generally, they proved that given a container with balls we can do any kind of computation.

The billiard ball model was taken further when Norman Margolus invented an implementation of a model cellular automaton. He published his result in 1984. "Margolus's" and "the billiard ball model's cellular automata" are widely used today.

The story I told you is just one of many possible explanations about how the field of collision-based computation emerged.

5.2. Definition

After reporting the background that gave rise to the collision-based computation, we could infer an initial definition of what this new computer model is and how it works, writing that "Collision-based computing is a model that allows us to perform operations similar to those of a computer using elements that when interacting between them generate the bases to compute something", is similar to our definition of computing, with the great difference that now our basic unit of information They will not be 0's and 1's, but they will be elements that will move through space.

These elements can be particles, billiard balls, marbles, electromagnetic waves, etc., and our intuition of '0 and 1 'could be handled as absence / presence, state e_1 and e_2, etc. This depends on the model and how we have defined it, the truth is that there is no correct way to make computation based on collisions, but there are ways that exploit more the computational capabilities of one system than another.

Thus, we can define the computation based on collisions in the following way:

It is a model that allows us to write algorithms that will work with basic units called mobile elements, which depending on their status, or their absence or presence, will determine their values equivalent to the binary system.

5.2.1. Main characteristics

Within the main characteristics of collision-based computation, we will mention the most common within the models found in the literature:

1. **It has a collision space:** Although we have elements that exist within our model and that interact with others, we need to define the physical space where these interactions will happen, it could be a pool table or the computational space of an ECA.

2. **It has mobile elements:** These mobile elements have their own characteristics, and also their properties emerge when they collide with others, the property they will share with others is that they must move.

3. **It is possible to interact these elements:** It will seem redundant, but we could find elements that are isolated and that we can not interact, which is

why this condition is important.

4. **The result of the interactions exists:** We could find the case that interacting causes them to be destroyed, so if the collision between several elements generates new ones, some are destroyed or simply bounce, this will allow us to model.

5.2.2. Objectives

Collision-based computation can have different objectives depending on where it is focused and what is to be resolved, but we can highlight the purpose for which this new form of computing emerged:

"Design a more efficient, simple and creative way to perform computation through interactions between elements within a space that exists in a complex system."

From here we can derive other particular objectives:

1. Optimize the number of operations when implementing conservative computation.

2. Allow the existence of the reversible computation to show all the steps that were followed to reach the result.

3. Optimize the way in which operations are performed by coding them in a different way.

4. Mapping the model to a physical model that allows us to design optimal computers and make better use of their resources.

5.3. Determining the computing power of a system

To determine the computational power of a system it is necessary to model some computation using the interactions between the elements that emerge from a complex system, this can be done by equations of collisions or equivalences between them. Once we have the model, depending on the quantity and type of operations that we

can perform with it, it is the associated computing power. If we can make many logic gates type AND and NOT, but with another model we can only make the NAND and NOR, the second model would become more powerful because the NAND and NOR gates are universal.

We will deal in depth with how the computing power of a system is determined using two main parameters: the Chomsky Hierarchy and the Logic Gates.

5.3.1. Chomsky Hierarchy

The Chomsky hierarchy in a classification of formal grammars that generate regular languages and allows us to identify which language and with which machine we can recognize it. This consists of 4 numbered types from 0 to 3, where 0 is the largest covering all the grammars, and contains 1, 2 and 3. Let's see the hierarchy graphically and then we will explain it one by one the types it has and its relation to the determination of the computing power of a system. In the figure 5.1 we can see an image with the 4 types of languages and the machine that recognizes them.

We can see that type 0 contains the rest of the languages, type 1 contains 2 and 3, 2 only 3, and 3 is the most basic. We will describe in detail each of these types:

1. **Type 3:** Known as regular languages, are those that can be expressed by a regular expression and can be linear on the right or on the left. This language recognizes sequences of symbols. They are recognized using finite automata, whether deterministic or non-deterministic.

2. **Type 2:** Know as context-independent languages, they are sets of sequences of symbols or phrases. These languages are recognized using stack automata.

3. **Type 1:** Known as context-dependent languages, they are sets of sets of symbol sequences. They are recognized using linearly coupled automata.

4. **Type 0:** Known as recursive languages, it is the largest hierarchy according to Chomsky and includes a set of formal objects of any computational complexity. They are recognized using Turing machines.

Knowing this, imagine that we have a system that allows collisions, and these can be modeled with equations, equivalences between states and between characters

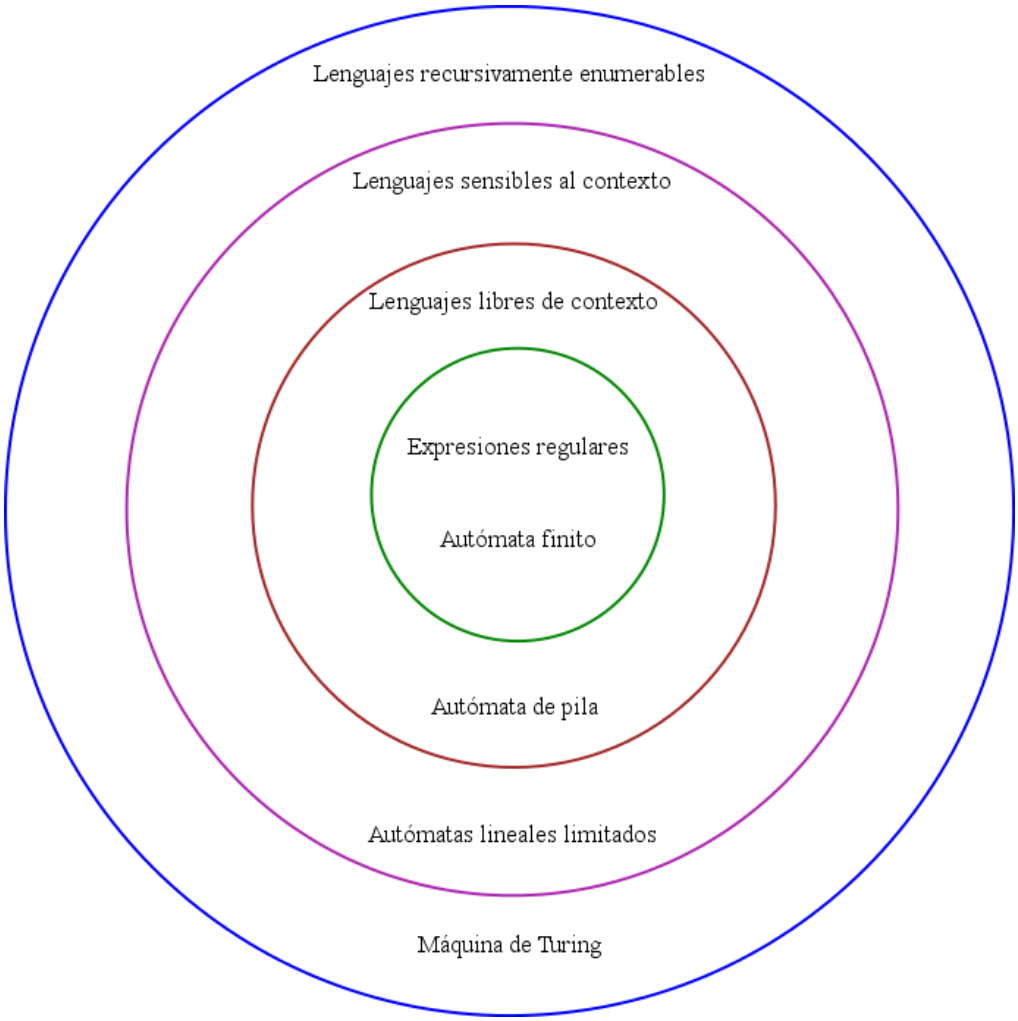


Figura 5.1: Chomsky Hierarchy

of an alphabet, etc. To measure the computational power of such a system, a good way is to start with Chomsky's hierarchy in type 3 and see if we can simulate a regular language using collisions, if we succeed we can move to type 2 and simulate a free grammar of context, if so we can pass type 1 and so on until we reach type 0 (perhaps the most complex), and try to simulate a Turing machine, depending on how far we can simulate, is the computing power we have found from our system.

5.3.2. Logic Gates

Logic gates are electronic circuits internally conformed by transistors that are in special arrangements with which they give voltage signals as a result or a Boolean output, this output can be a high voltage (eg 5V = 1 in binary), or a low voltage (eg 0V = 0 in binary). With these logic gates we can compute everything we know, in fact our processor is based on an array of transistors which has a lot of gates and when receiving an instruction transforms it into voltage, passes it through the transistors and returns the result.

For logic gates we can handle two types of logic:

1. **Positive logic:** This means that when it receives a signal in high it is activated, representing a 1 in binary, and with a low it is deactivated, representing a 0 in binary.

2. **Negative logic:** This is the opposite of the positive logic, when it receives a low the signal is activated, represented with a binary 0 and with a high it is deactivated, represented with a 1 in binary.

This logic can also be expressed through the boole algebra, which establishes the rules for multiplication and sum of entries according to the gate that is being applied. There are many logical gates, but the basic (or elementary) are the following:

AND Gate

This logic gate evaluates two input signals and as a result it throws a 1 if both signals are high, otherwise it throws 0. In a simple way, taking the signals a and b, we can say that we are going to do the comparison "a Y b", or a multiplication within the boolean algebra. The truth table of this gate is in the table 5.1.

a	b	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

Cuadro 5.1: AND gate truth table

OR Gate

This logic gate evaluates two input signals and as a result it throws a 1 if any of the signals is high, otherwise it throws 0. Say it in a simple way, taking the signals a y b, we can say that we are going to make the comparison "a O b", or a sum within the boolean algebra. The truth table of this gate is in the table 5.2.

a	b	a OR b
0	0	0
0	1	1
1	0	1
1	1	1

Cuadro 5.2: OR gate truth table

NOT Gate

This logic gate evaluates an input signal and as a result throws the complement to 1 of the signal. Simply put, taking the signal a, we can say that we are going to make the denial "NO a", within the boolean algebra a line is placed on top of the signal. The truth table of this gate is in the table 5.3.

a	NOT a
0	1
1	0

Cuadro 5.3: NOT gate truth table

NAND Gate

This logic gate is similar to the AND but denied, evaluates two input signals and as a result throws a 1 if the signals are different from being high, otherwise throws a 0. Simply put, taking the signals a y b, we can say that we are going to make the comparison "NO a Y b", or a sum within the boolean algebra and then the result to apply a NOT. The truth table of this gate is in the table 5.4.

a	b	a OR b
0	0	1
0	1	1
1	0	1
1	1	0

Cuadro 5.4: NAND gate truth table

NOR Gate

This logic gate is similar to the OR but negated, evaluates two input signals and as a result throws a 1 if both signals are low, otherwise it throws 0. Putting this simple, taking the signals a y b, we can say that we are going to make the comparison "NO a O b", or a multiplication within the boolean algebra and then the result to apply a NOT. The truth table of this gate is in the table 5.5.

a	b	a NOR b
0	0	1
0	1	0
1	0	0
1	1	0

Cuadro 5.5: NOR gate truth table

XOR Gate

This logic gate is similar to the OR but is known as exclusive OR, evaluates two input signals and throws a 1 if both signals are different, otherwise it throws 0. Put

simply, taking the signals a y b, we can say that we are going to make the comparison " (a O NO b) Y (NO a O b)", or the sum of the multiplications of a with b denied and vice versa. The truth table of this gate is in the table 5.6.

a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

Cuadro 5.6: XOR gate truth table

XNOR Gate

This logic gate is similar to the OR but it is known as Exclusive OR denied, evaluates two input signals and throws a 1 if both signals are equal, otherwise it throws 0. Say it in a simple way, taking the signals a y b, we can say that we are going to make the comparison " NO a XOR b", or the sum of the multiplications of a with b and the denial of the other multiplication. The truth table of this gate is in the table 5.7.

a	b	a XNOR b
0	0	1
0	1	0
1	0	0
1	1	1

Cuadro 5.7: XNOR gate truth table

IF Gate

This logical gateway seems to do no harm if we only see the logical part. If it receives a 1 it throws a 1, and if it receives a 0 it throws a 0, but in reality it is a current amplifier, it amplifies the impedance of our signal. The truth table of this gate is in the table 5.8.

a	IF a
0	0
1	1

Cuadro 5.8: IF gate truth table

Finally, we can see all the logic gates graphically along with their equation according to the boole algebra in the figure 5.2

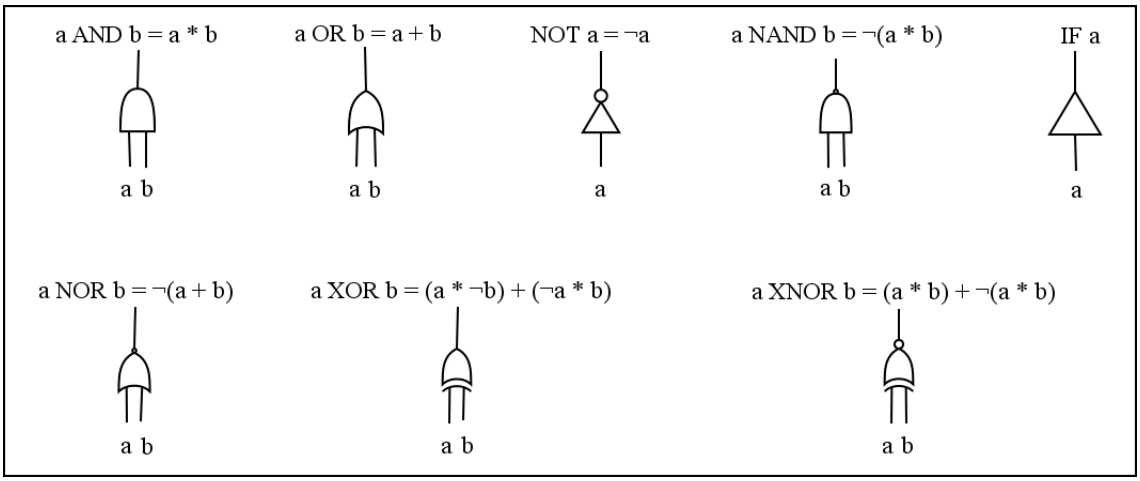


Figura 5.2: The basic logic gates

The universality of logic gates

We have seen the basic logic gates, but why are they known as basic? There are very many circuit arrangements that allow us to do more complex and more interesting things like flip-flops, registers, barrel shifters, etc., but all these circuits, however interesting they appear internally, have 3 of the logic gates described above: AND, OR and NOT, and is that with them we can build all the logic that we require to compute anything, even if we only take the NAND and NOR, we get the same result since these two gates contain the NOT. These gates are known as "universal", and if a system has them, it can perform universal computation, or it has an associated Turing machine.

Capítulo 6

Constructions based on collisions

6.1. Objects of rule 126 with memory

So far we have seen collisions involving two or three particles within the space of evolutions, these collisions allowed us to explore a bit the behavior of the rule. However, as it happens in our universe, although it is required to know the interaction between two elements, our world works with thousands of particles, which together generate more complex results and are what we perceive. In this section we will describe some results generated by rule 126 with memory when we add many particles within the space of evolutions, which we will label according to their collective behavior.

6.1.1. Eaters

They are structures that eat gliders. The eaters are usually also gliders that feed on others, retain their shape but move some cells either to the left or to the right. In the figure ref fig: eaters we can see some eaters that have the rule 126 with memory.

In the figure 6.1, we can observe 4 examples of eaters, which we will describe below:

a) This is an eater that disperses both left and right, and travels at two different speeds:

- The first one traveling to the left has a speed of $V_{izq} = \frac{3}{19} = 0{,}158$

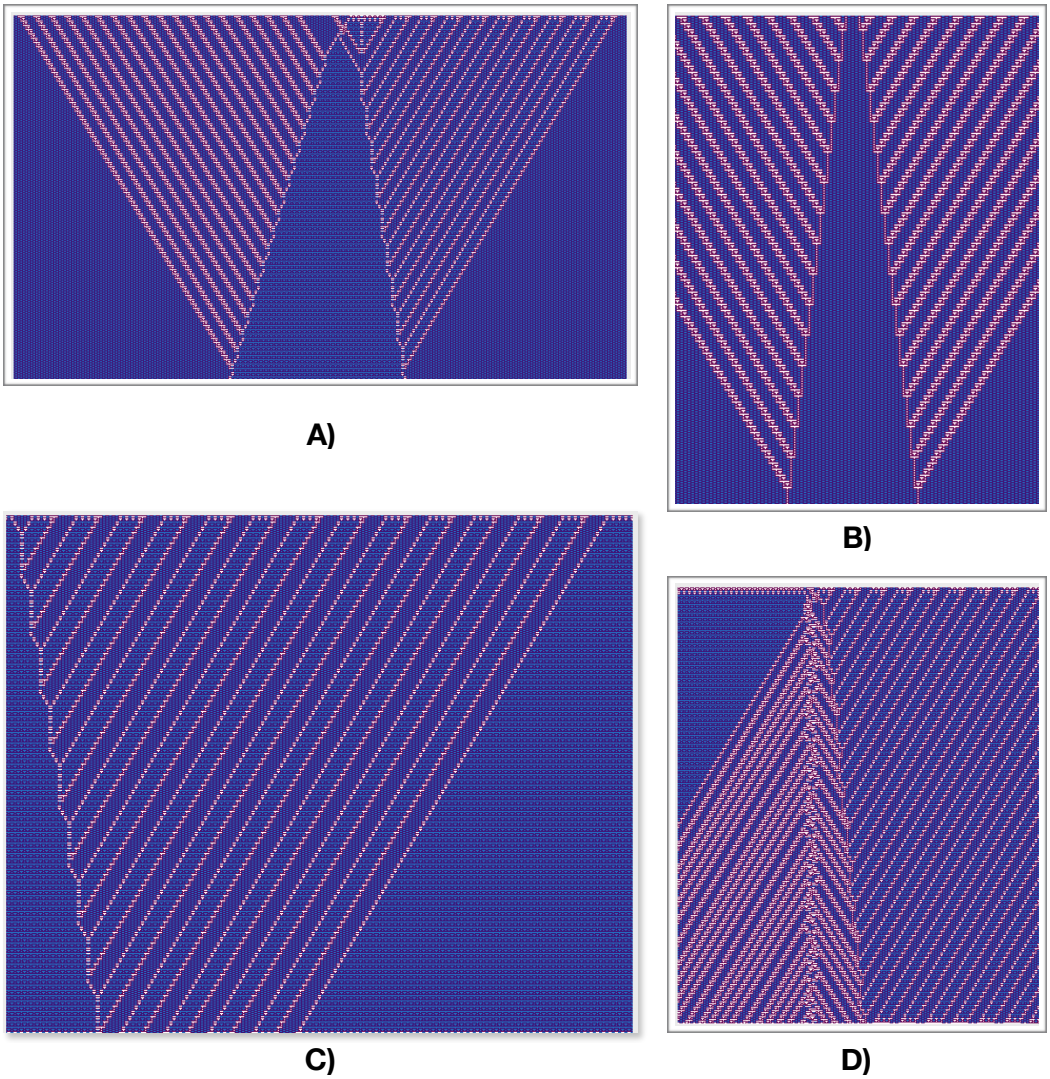


Figura 6.1: Some examples of eaters

- The second one that travels to the right has a velocity of $V_{der} = -\frac{3}{39} = -0{,}077$

b) This eater on the contrary, handles equal speeds on both the left and the right, which is $V = \frac{2}{25} = \pm 0{,}08$

c) The 3rd eater combines the two tiles for filtering at the same time that it moves from left to right, with a speed $V = -\frac{6}{55} = -0{,}109$

d) Finally, the 4th eater, which is a glider-gun that is eating gliders on its right side as it advances, has a velocity of $V = -\frac{12}{100} = -0{,}12$

6.1.2. Black-holes

They are structures that eat gliders and maintain their shape, the difference is that they do not move while they are doing it, no change of position is perceived in them. In the figure 6.2 we can see an example of a black hole.

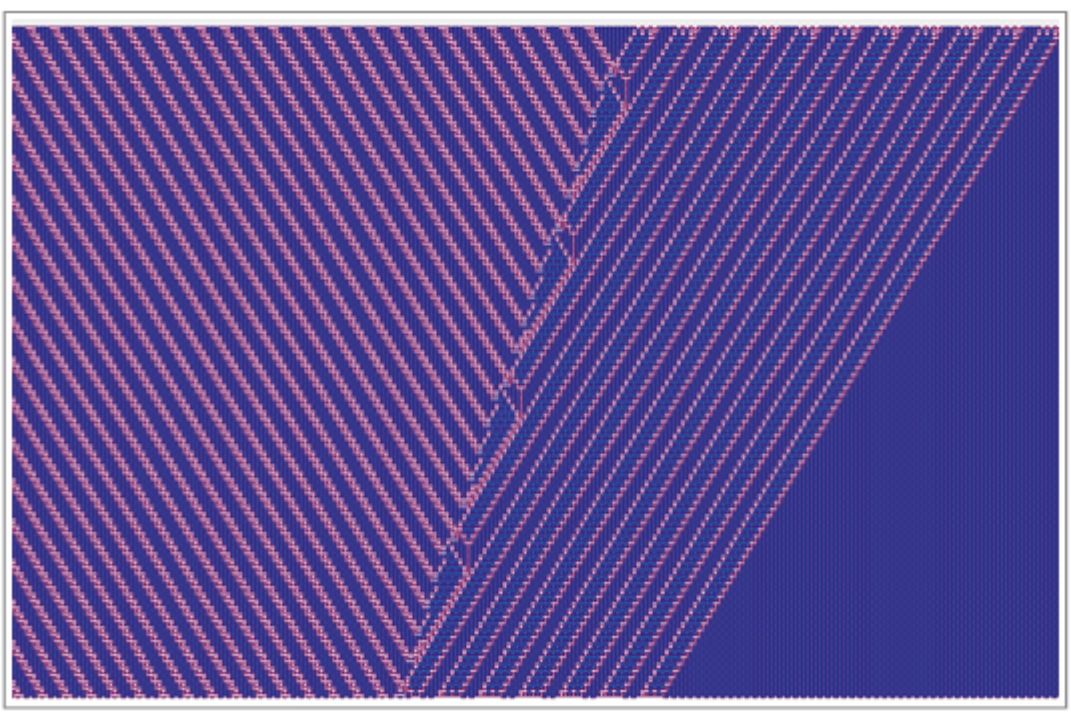


Figura 6.2: An example of a black hole

The black hole of the figure 6.2 has a fairly large period, although its speed is minimal. The interesting part of this black hole is that it consumes different gliders on the left and on the right, being an example of complexity within the rule.

6.1.3. Solitons

Solitons are structures that allow other gliders to pass through them without affecting them or affecting the gliders. Within the study of signals, waves that behave like solitons play a very important role in communications. In the figure 6.3 we can observe two solitons of the rule 126, one that works in the tile for filtering 1 and the other in the 2nd tile for filtering.

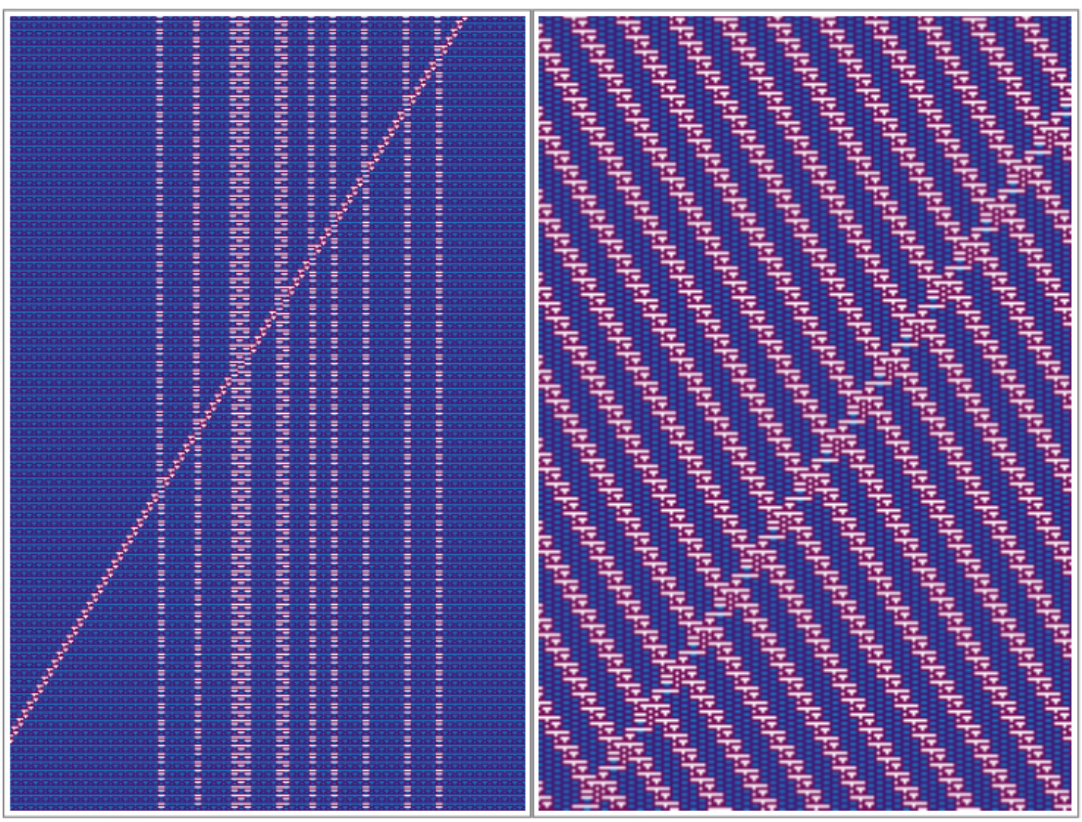


Figura 6.3: Some examples of solitons

In the figure 6.3, we can observe that both solitons move to the left, however one of them allows the passage of gliders and the other of still-life, so we can design

records using them.

6.2. Computability of the rule

Now that we know the basic concepts of the previous section, we are going to explore the computing power of rule 126 with memory of 4 generations, we have all the tools we need: we have collided relevant binary and ternary combinations and we have the equations that will allow us to gate gates or from the Chomsky Hierarchy. In this section we will show the results and the conclusions we reached from the exploration.

6.2.1. Regular languages

A regular language occupes type 3 within the hierarchy of Comsky and is the most elementary when it comes to describing languages and recognizing chains that belong to it. We will describe a regular language using particle collisions.

Language $(01)^n$

We define the following:

1. Be the alphabet $\Sigma = \{0, 1\}$
2. Be the empty string $\epsilon = ""$
3. We define the following regular language:
   - $L = \{(0, 1)^n | n \geq 0\}$
4. Some strings that belong to this language are the following: $\{\epsilon, 01, 0101, 010101, 01010101, ...\}$

From this definition we can obtain its regular expression, cited in the equation 6.1

$$RE = (01)^* \tag{6.1}$$

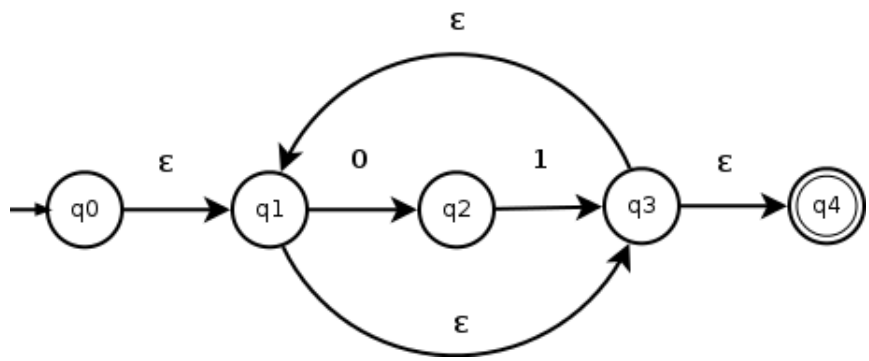


Figura 6.4: Non-deterministic Finite automaton of regular language $(01)^*$

If we graph the non-deterministic finite automaton of this regular expression we get the figure 6.4

If we simplify the automaton [14] we can get the figure 6.5

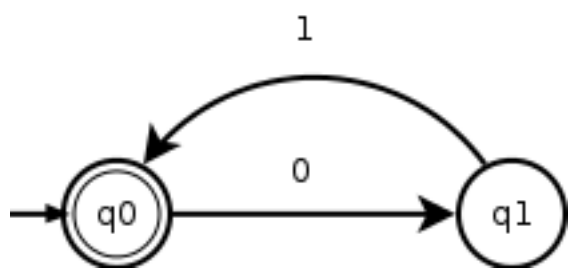


Figura 6.5: Non-deterministic Finite automaton simplified $(01)^*$

We can get the transitions table, which is in the table 6.1

State	0	1
$\rightarrow *q_0$	q_1	$\emptyset$
q_1	$\emptyset$	q_0

Cuadro 6.1: Table of transitions of the NFA of the language $(01)^*$

Now we are going to do two types of equivalences: the character-particle equivalences and the state-particle equivalences. For the first ones we have that:

1. The glider g_4 will be the character 0

2. The glider g_6 will be the character 1

For state-particle equivalencies, we have that:

1. The still-life s_2 will be equal to the state q_0
2. The glider g_5 will be equal to the state q_1

Using these equivalences, we can modify the figure 6.5 with the new notation, which is in the figure 6.6, as we can see, now the states are labeled by particles and transitions by others.

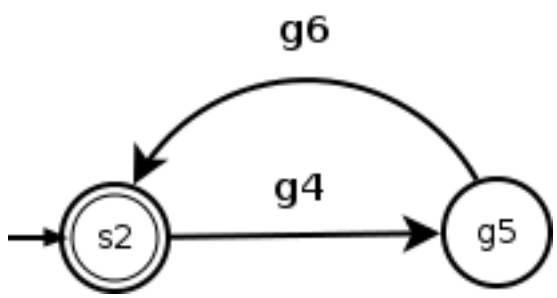


Figura 6.6: Non-deterministic Finite Automaton of the regular language $(01)^*$ with the new equivalences

In the same way we can make the change in the table of transitions, which would be as shown in the table 6.2.

Estado	g_4	g_6
$\rightarrow *s_2$	g_5	$\emptyset$
g_5	$\emptyset$	s_2

Cuadro 6.2: Table of transitions of the NFA of the language $(01)^*$ with particles

Now let's observe how this regular language works with collisions between particles when we want to recognize certain chains:

For the empty string: If we have input the empty string ϵ, we have that there will be no transitions within the automaton, however it always starts in the initial

state s_2 and since it is also the final state, it will be a chain accepted by the language. In the figure 6.7 we can observe the space of evolutions and the null alteration of this one.

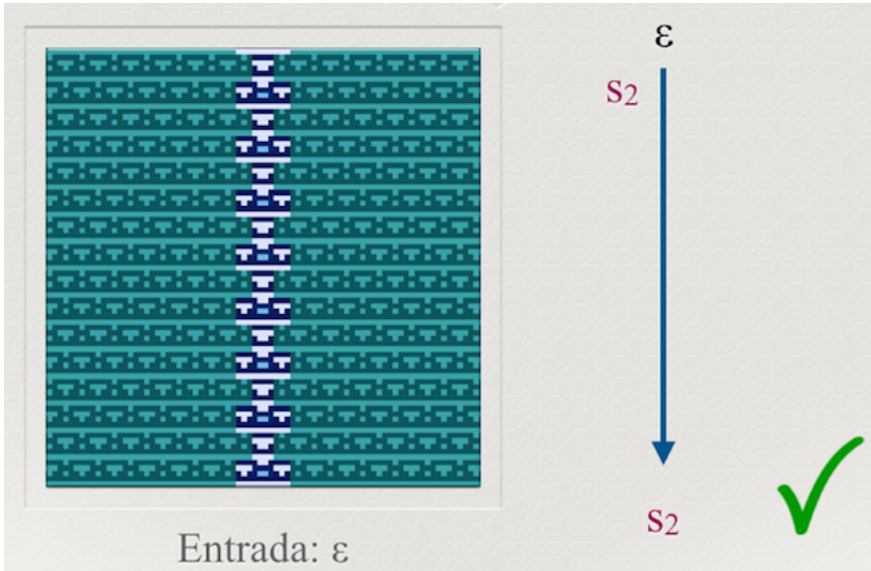


Figura 6.7: Evaluation of the empty string ϵ

For a zero as input: If now we enter a 0 and evaluate it with the automaton, we would start in the state s_2 and according to our new table of transitions, we would move to g_5 when we receive g_4 (the equivalence of 0) as input, however g_5 is not a final state, so the string 0 would not be accepted by the language. In the figure 6.8 we can observe this behavior in the space of evolutions.

For a zero and one as input: If we have input 01 and evaluate it with the automaton, we would start in the state s_2 and we would move to g_5 when reading $g_4(0)$, and then because we are in g_5 and we receive a g_6 (equivalence of 1), we return to s_2, and s_2 is our final state, the string would be accepted by the language. In the figure 6.9 we can observe this behavior in the space of evolutions.

A larger string(010101010): Now we will use a somewhat large string, the string 010101010, although it seems that it will be accepted by the language, because it ends with a 0 our automaton will reject it. In the figure 6.10 we can observe how it evolves with the previously explained and the result that is a false one.

A larger string(01010101): We will make an evaluation similar to the previous

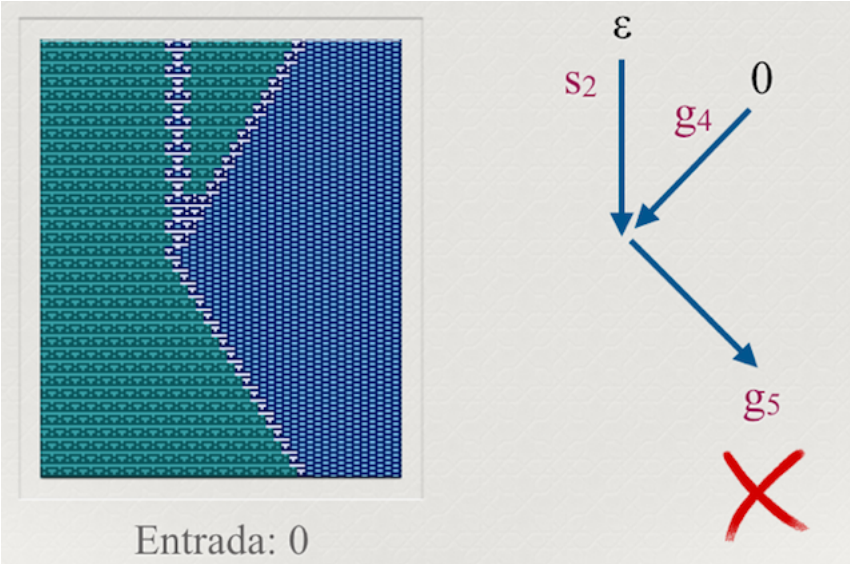


Figura 6.8: Evaluation of the string 0

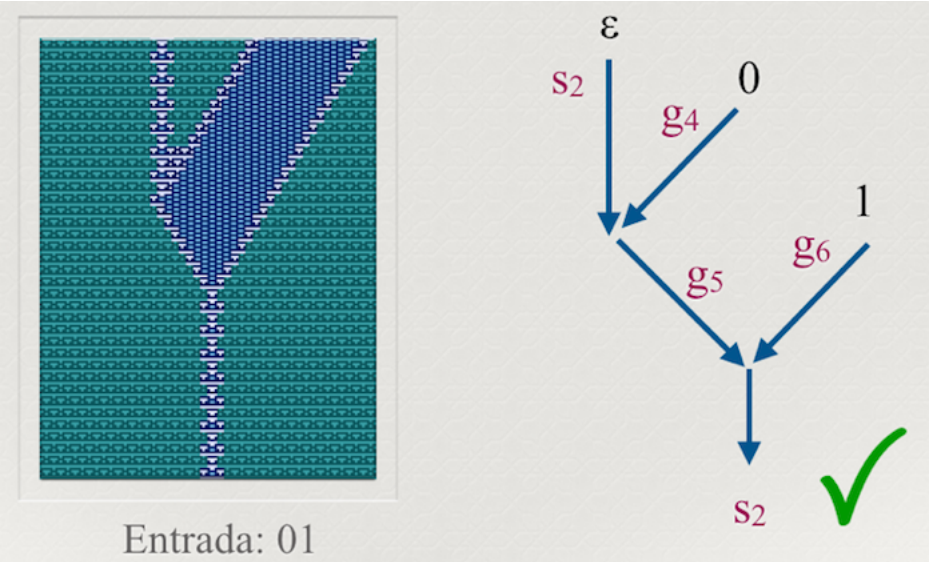


Figura 6.9: Evaluation of the string 01

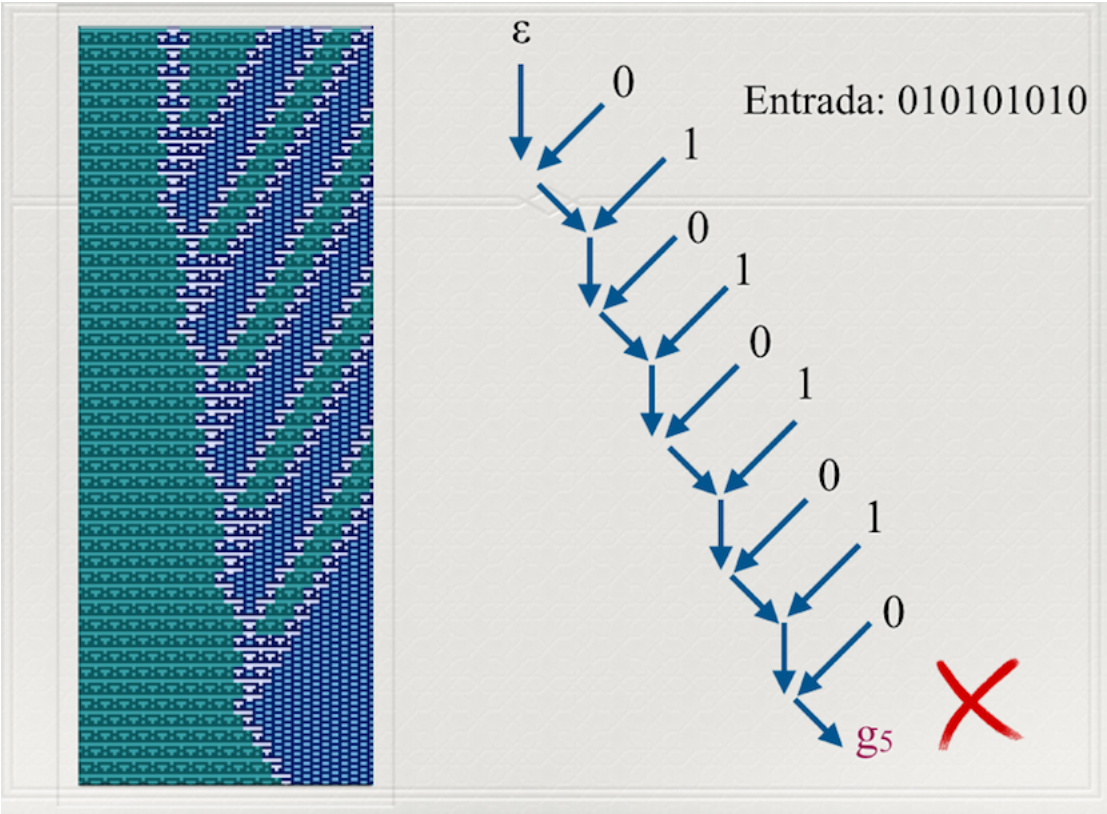


Figura 6.10: Evaluation of the string 010101010

one, only that now we will remove the 0 from the end of the chain, thus it would be a string that belongs to the language that we define. In the figure 6.11 we can observe the evolution.

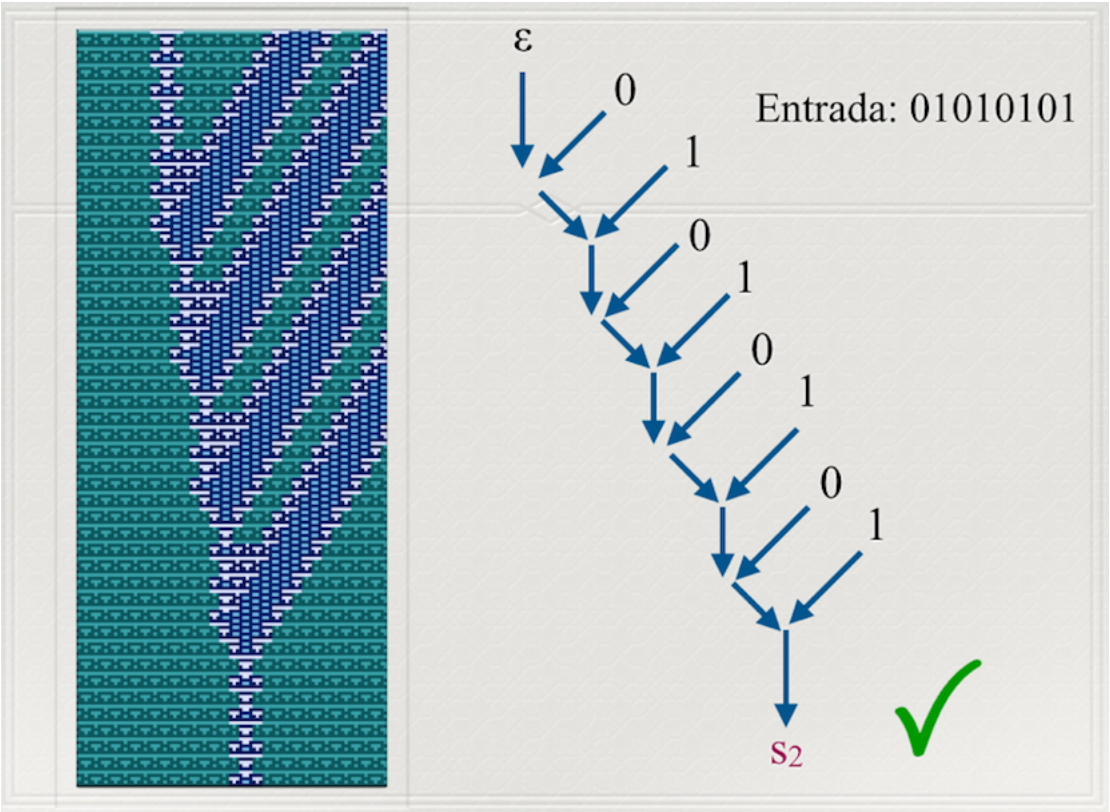


Figura 6.11: Evaluation of the string 01010101

This is an example of regular language design, we could change the equivalences of the characters and now use letters like a y b.

6.2.2. Context-Free Grammars

Context-free grammars belong to type 2 within the Chomsky hierarchy and define production rules to generate strings that belong to the language, in the same way they allow to recognize strings like type 3. In this section we will exemplify a context-free grammar.

Language $(01)^n$

We will take the language defined in the section 6.2.1 and we will define its context-free grammar in the following way:

1. $G = (\{A\}, \{0, 1, \lambda\}, P, S)$, where P contains the following productions:
   - $S \to A$
   - $A \to 01A$
   - $A \to \lambda$

Now, we will make equivalences of the definition of our grammar with the particles in the following way:

1. $G = (\{g_5\}, \{g_4, g_6, s_2\}, P, S)$, where P contains the following productions:
   - $S \to g_5$
   - $g_5 \to g_4 g_6 g_5$, which is obtained when making the collision: $g_5 \leftrightarrow g_2 + g_2$
   - $g_5 \to s_2$, which is obtained when making the collision: $g_5 \leftrightarrow g_6$

We are going to make an example using this grammar, what we want is to generate the string 010101 using our production rules and starting with S. First you would change to S by g_5, and then we need to apply 3 times the production $g_5 \to g_4 g_6 g_5$ being that $g_4 = 0$ and $g_6 = 1$, and in turn returns us g_5 which is our non-terminal, this production we get if we collide g_5 against $g_2 + g_2$, then to generate the empty string $s_2 = \lambda$, we collide g_5 with g_6 and we get our string. In the figure 6.12 we can observe this behavior in the space of evolutions and the equivalences with the grammar.

In this way, we can define context-free grammars that allow us to generate chains through the derivation rules, which will be controlled with the collisions of the particles.

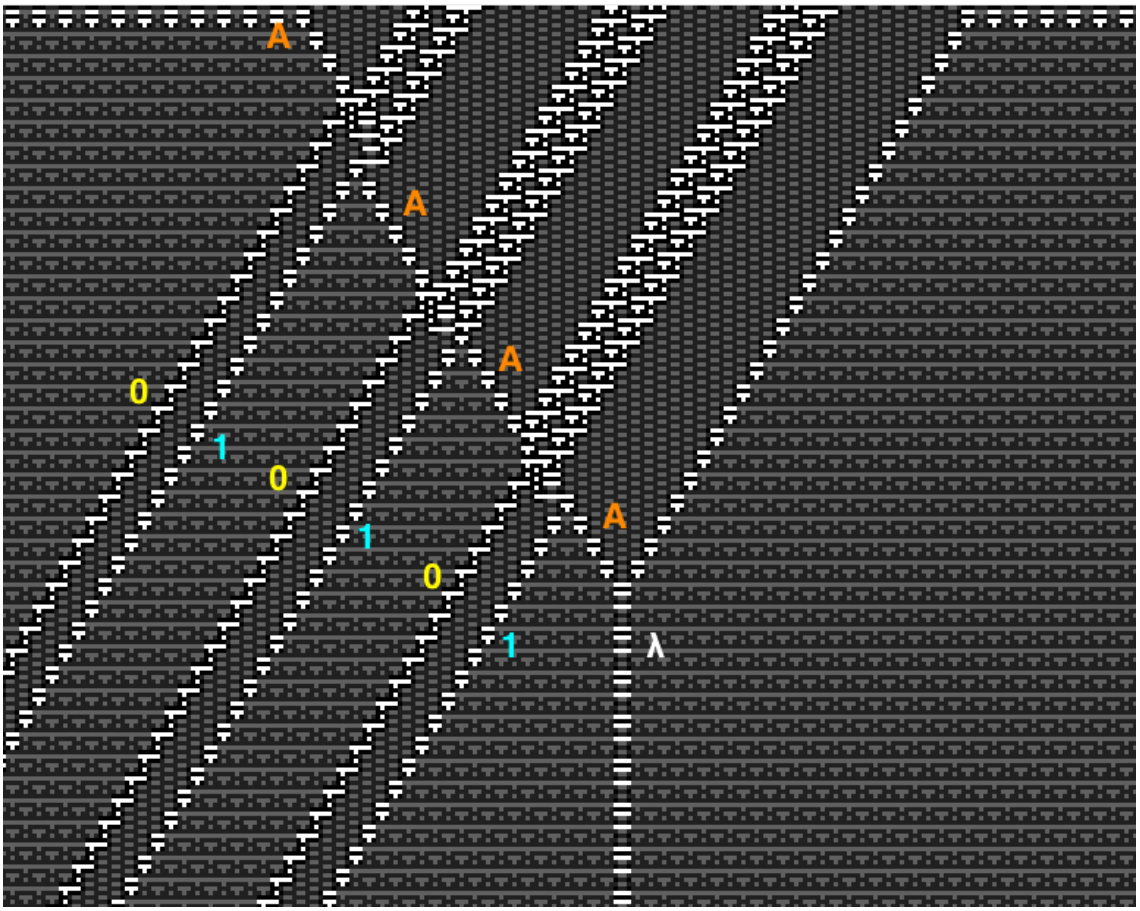


Figura 6.12: Generation of the branch tree for the string 010101

6.2.3. Logic gates

Logic gates, as described in the chapter 5, allow to perform logical operations between one or more bits, resulting in a bit that will depend on its value from a truth table. We will take the logic gates described in this chapter and try to simulate their behavior with particle collisions.

XOR Gate

The XOR gate is defined in the section 5.3.2 and we can say in a general way that if it receives equal entries, then the result will be 0, otherwise it will be a 1. Let's model this gate using the particles g_1, g_1^2 and g_2 as follows:

1. g_1 will be our entrance "a" of the XOR gate, and when this input is equal to 0, it means that the particle is not present in the space of evolutions, when it is equal to 1, it implies that it is in space.

2. g_1^2 will be our entrance "b" of the XOR gate, and when this input is equal to 0, it means that the particle is not present in the space of evolutions, when it is equal to 1, it implies that it is in space.

3. g_2 will act as our XOR operator, when we apply it to the tickets it will do the operation and it will return the result according to what it received in the first place.

In the table 6.3, we can see the truth table of the XOR gate with the new equivalences, as well as the equation that describes it and the result that we obtain.

g_1	g_1^2	g_1 XOR g_1^2	Equation	Result
0	0	0	g_2	g_2
0	1	1	$g_1^2 + g_2$	g_1^2
1	0	1	$g_1 + g_2$	g_1
1	1	0	$g_1 + g_1^2 + g_2$	$\emptyset$

Cuadro 6.3: Truth table of the XOR gate with particles

In the figure 6.13 we can observe how the particles behave when we evaluate the 4 different possibilities, taking the glider g_2 as our XOR operator, and we can see that they correspond to those defined within the truth table.

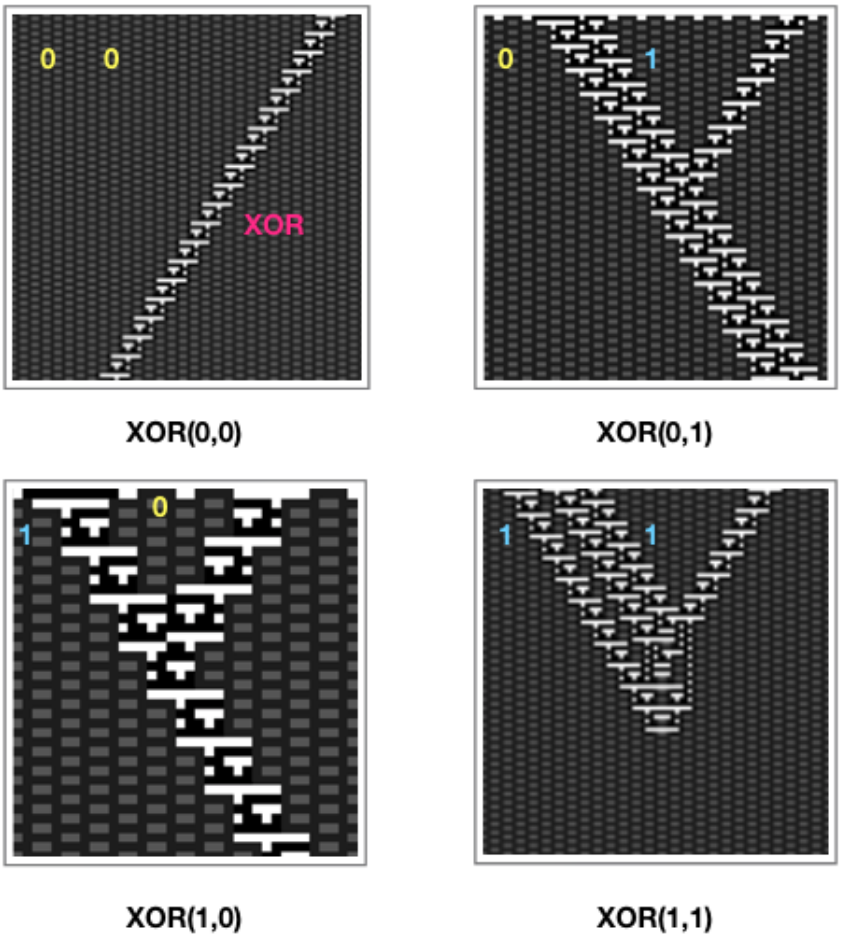


Figura 6.13: The 4 combinations of the truth table for the XOR gate

Conditional IF-THEN

The conditional IF-THEN is a gateway that evaluates the sentence: if A then B, and its truth table tells us that only if it receives a 0 in the first entry and a 1 in the second, the output will be 0, otherwise It will be a 1. We can model this gate using the same scheme as the NAND but changing the operator.

1. g_1 will be our equivalent to the value 1 in binary.
2. g_1^2 will be our equivalent to the value 0 in binary.
3. $g_2 + g_2$ will act as our IF-THEN operator, when we apply it to the tickets it will do the operation and it will return the result according to what it received in the first place.

So, in the table 6.4, we can observe the truth table of the IF-THEN gate and its equivalent to the particle equations. We can see that the equations are very similar to the NAND, only that our IF-THEN operator is $g_2 + g_2$.

a	b	IF a THEN b	Equation	Result
0	0	1	$g_1^2 + g_1^2 + g_2 + g_2$	g_1
0	1	0	$g_1^2 + g_1 + g_2 + g_2$	g_1^2
1	0	1	$g_1 + g_1^2 + g_2 + g_2$	g_1
1	1	1	$g_1 + g_1 + g_2 + g_2$	g_1

Cuadro 6.4: Truth table of the IF-THEN gate with particles

In the figure 6.14 we can observe the 4 combinations within the evolutions space, which indicate that we can effectively simulate the behavior of the gate with the collisions, this gate together with the NAND work with the same scheme of equivalences between particles, so we can use them together to generate larger circuits.

NAND gate

The NAND gate is defined in the section 5.3.2 and we can say in a general way that if it receives both inputs as a 1 then it will return a 0, otherwise it will return a 1. Let's model this gate using the particles g_1, g_1^2 and g_2^2 as follows:

1. g_1 will be our equivalent to the value 1 in binary.
2. g_1^2 will be our equivalent to the value 0 in binary.
3. g_2^2 will act as our NAND operator, when we apply it to the inputs it will do the operation and it will return the result according to what it received in the first place.

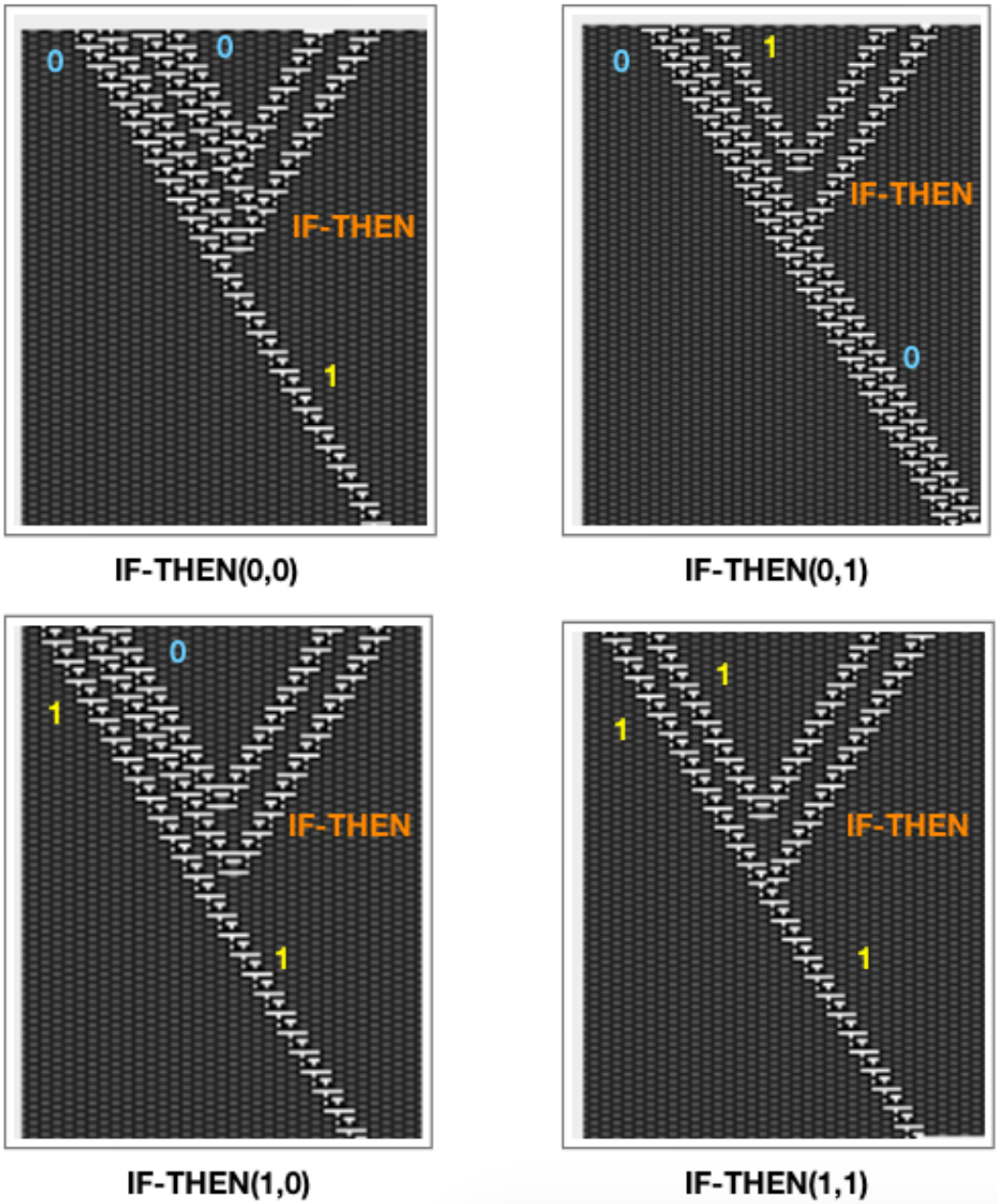


Figura 6.14: The 4 combinations of the truth table for the IF-THEN gate

In the table 6.5, we can see the truth table of the NAND gate with the new equivalences, as well as the equation that describes it and the result that we obtain.

a	b	a NAND b	Equation	Result
0	0	1	$g_1^2 + g_1^2 + g_2^2$	g_1
0	1	1	$g_1^2 + g_1 + g_2^2$	g_1
1	0	1	$g_1 + g_1^2 + g_2^2$	g_1
1	1	0	$g_1 + g_1 + g_2^2$	g_1^2

Cuadro 6.5: Truth table of the NAND gate with particles

In the figure 6.15 we can observe how the 4 possible combinations of the truth table of the NAND gate behave, and we can observe that there is consistency in both the input particles and the operator, and it returns the corresponding result. One of the implications of the fact that rule 126 with memory of 4 generations using a majority function allows us to model a NAND gate, is that because this gate is universal, we can say that the rule is **lógicamente universal**.

AND gate

The AND gate is defined in the section 5.3.2 and we can say in a general way that if it receives both inputs as a 1 then it will return a 1, otherwise it will return 0. We are going to model this gate using the NAND gate that we defined previously, for this we are going to use the NAND of the figure 6.16

In the table 6.6, we can see the truth table of the AND gate using pure NAND, as well as the result obtained.

a	b	a AND b	Equation	Result
0	0	0	$(g_1^2\ NAND\ g1^2)\ NAND\ (g1^2\ NAND\ g1^2)$	g_1^2
0	1	0	$(g_1^2\ NAND\ g1)\ NAND\ (g1^2\ NAND\ g1)$	g_1^2
1	0	0	$(g_1\ NAND\ g1^2)\ NAND\ (g1\ NAND\ g1^2)$	g_1^2
1	1	1	$(g_1\ NAND\ g1)\ NAND\ (g1\ NAND\ g1)$	g_1

Cuadro 6.6: Truth table of the AND gate which were designed using NAND

In the figure 6.17 we can observe how the 4 possible combinations of the truth

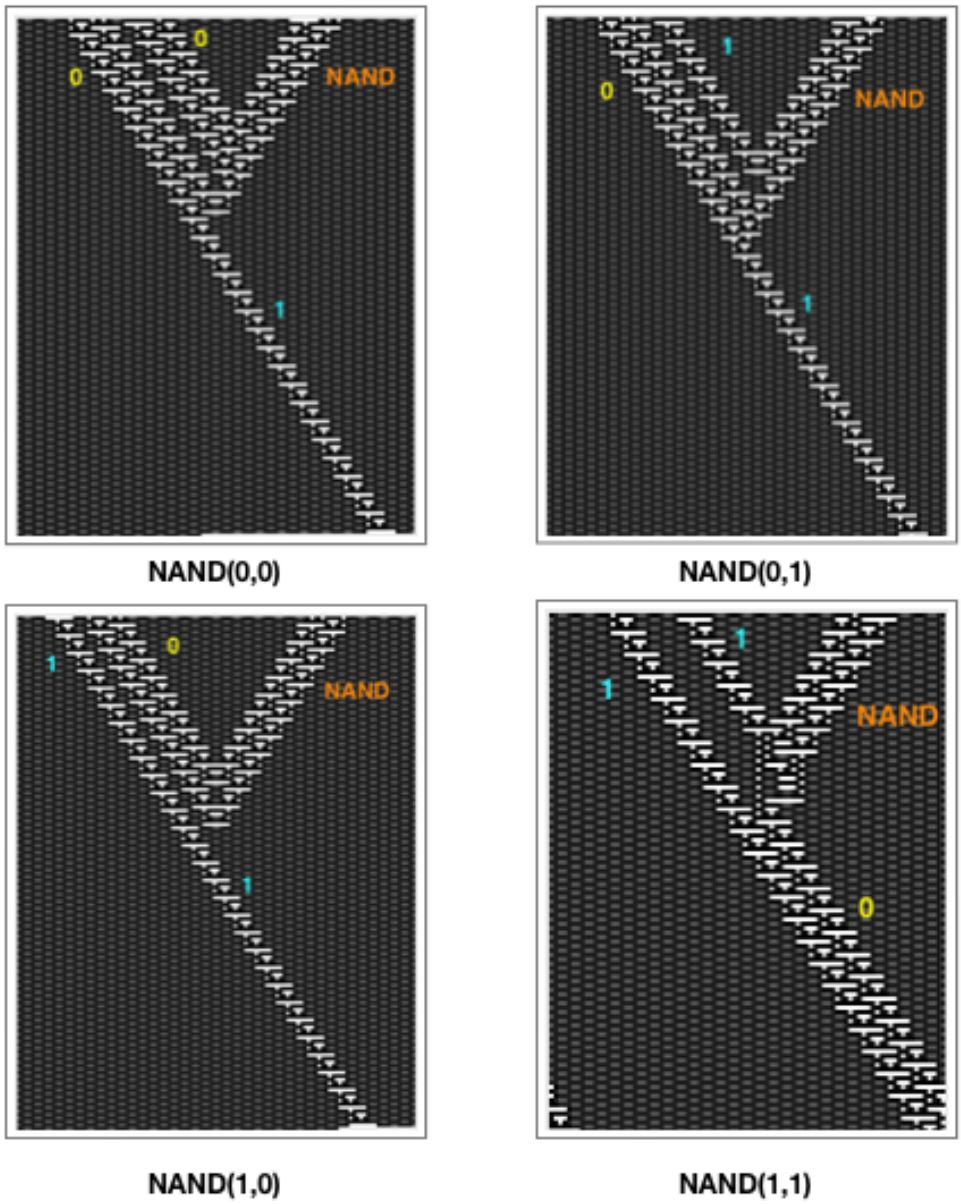


Figura 6.15: The 4 combinations of the truth table for the NAND gate

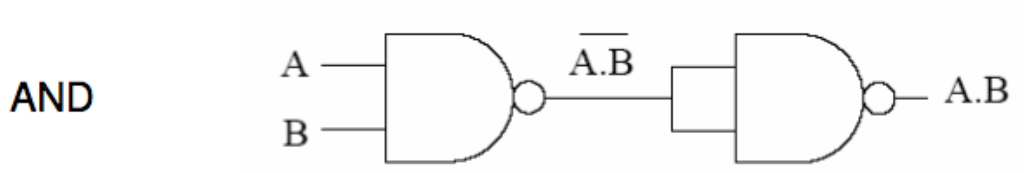


Figura 6.16: The AND gate built with NAND

table of the NAND gate behave, and we can observe that there is consistency in both the input particles and the operator, and returns the corresponding result.

OR gate

The OR gate is defined in the section 5.3.2 and we can say in a general way that if it receives any input that is in 1 then it will return a 1, otherwise it will return 0. We are going to model this gate using the NAND gate that we defined previously, for this we are going to use the NAND of the figure 6.18

In the table 6.7, we can see the truth table of the OR gate using pure NAND, as well as the result obtained.

a	b	a OR b	Equation	Result
0	0	0	$(g_1^2\ NAND\ g1^2)\ NAND\ (g1^2\ NAND\ g1^2)$	g_1^2
0	1	1	$(g_1^2\ NAND\ g1^2)\ NAND\ (g1\ NAND\ g1)$	g_1
1	0	1	$(g_1\ NAND\ g1)\ NAND\ (g1^2\ NAND\ g1^2)$	g_1
1	1	1	$(g_1\ NAND\ g1)\ NAND\ (g1\ NAND\ g1)$	g_1

Cuadro 6.7: Truth table of the OR gate which where designed using NAND

In the figure 6.19 we can observe how the 4 possible combinations of the truth table of the NAND gate behave, and we can observe that there is consistency in both the input particles and the operator, and it returns the corresponding result.

NOT gate

The NOT gate is defined in the section 5.3.2 and it is simply to invert the output we receive, so if we receive a 1 then a 0 will come out. Let's model this gate using

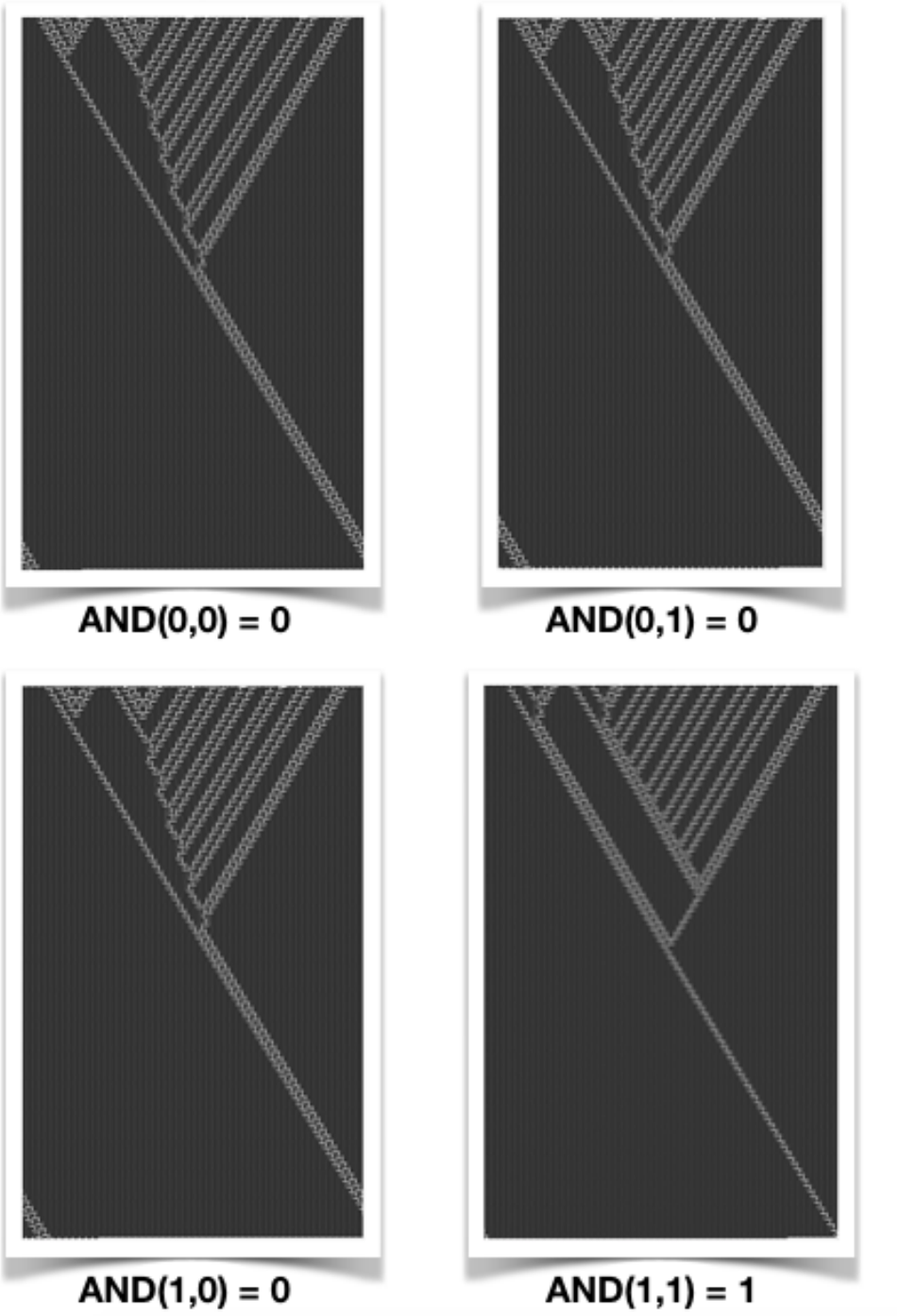


Figura 6.17: The 4 combinations of the truth table for the AND gate

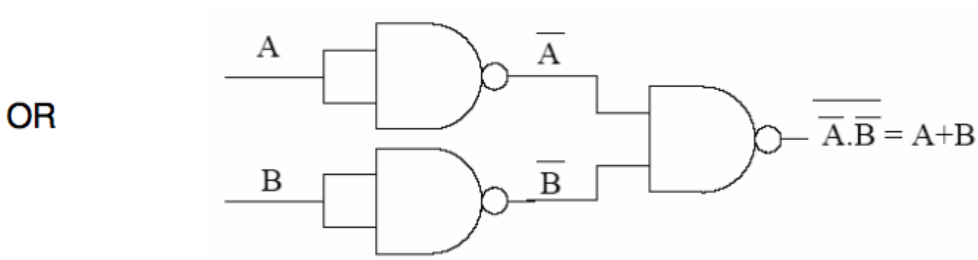


Figura 6.18: The OR gate built with NAND

the NAND gate that we defined previously, for this we will use the NAND of the figure 6.20

In the table 6.8, we can see the truth table of the NOT gate using pure NAND, as well as the result obtained.

a	NOT a	Equation	Result
0	1	$g_1^2 \ NAND \ g1^2$	g_1
1	0	$g1 \ NAND \ g1$	g_1^2

Cuadro 6.8: Truth table of the NOT gate with NAND

In the figure 6.21 we can observe how the 2 possible combinations of the truth table of the NOT gate behave, and we can observe that there is consistency in both the input particles and the operator, and it returns the corresponding result.

6.2.4. Rule computing power

Once we have finished the analysis of the possible systems that can be modeled using the collisions of the particles, we can make an opinion of what was found and what could possibly be in the rule:

1. **The rule is logically universal:** This fact is based on the simulation of the gate NAND using the particles, because this gate is one of the two universal gates (NAND and NOR), and that with them you can build all the other gates and equations according to algebra of boole, if we contain this gate then it allows us to say that the rule is logically universal.

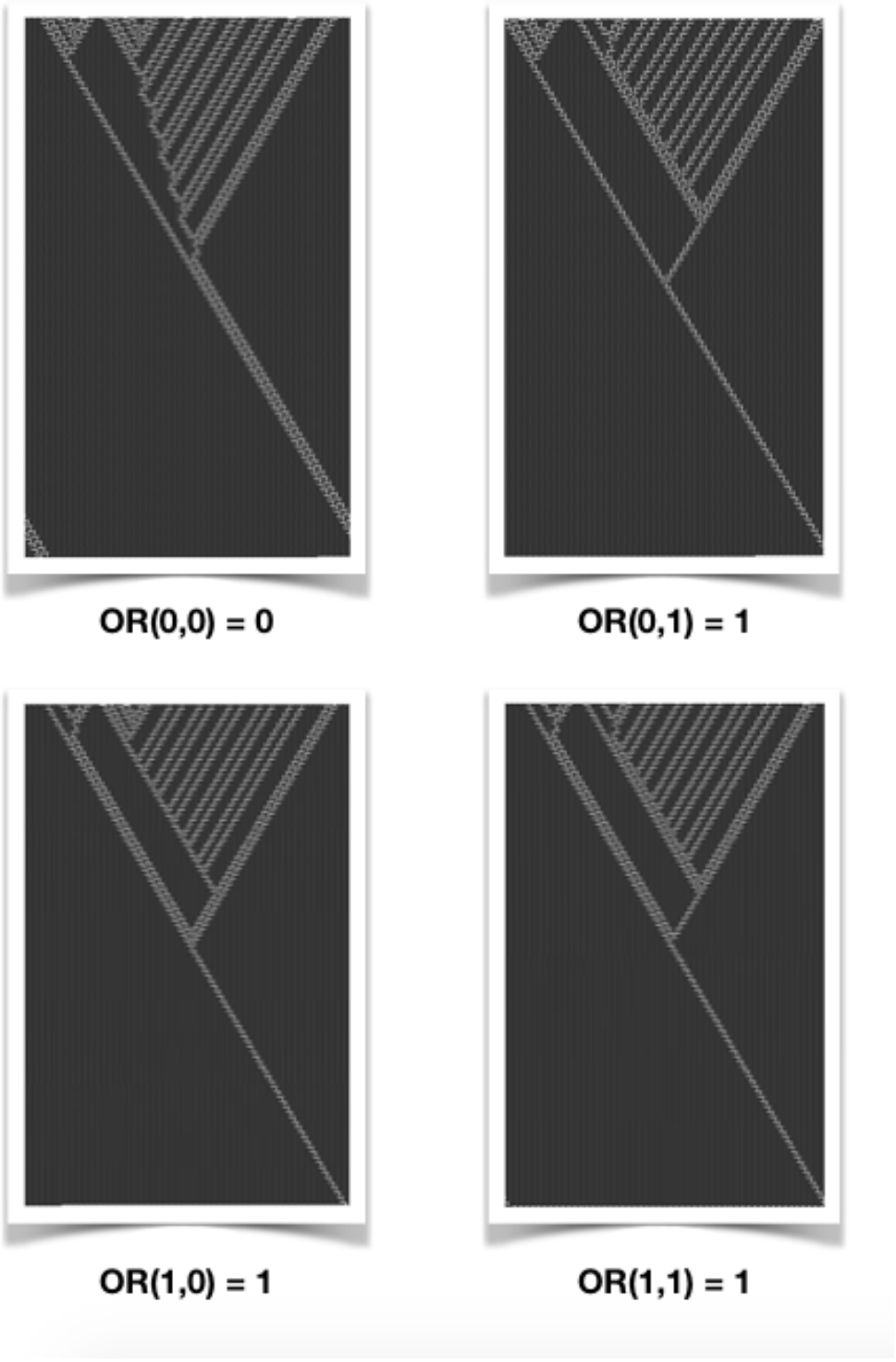


Figura 6.19: The 4 combinations of the truth table for the OR gate

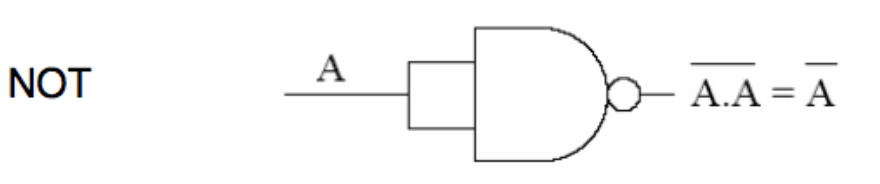


Figura 6.20: The NOT gate built based on NAND

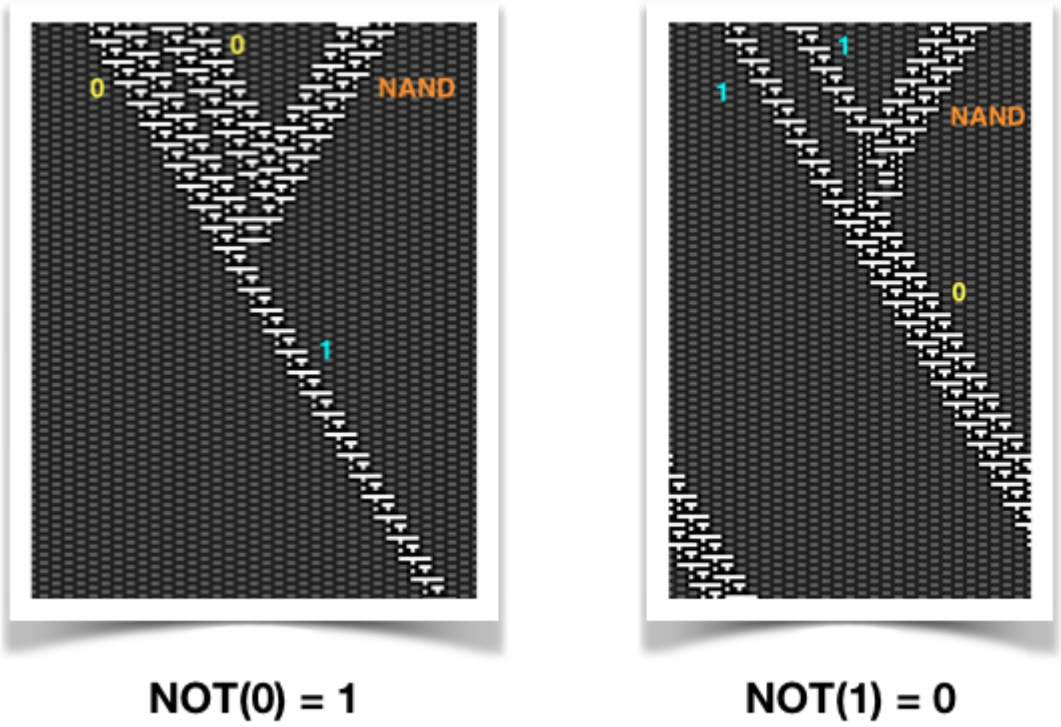


Figura 6.21: The 2 combinations of the truth table for the NOT gate

2. **The rule could have a Turing machine associated:** One of the most relevant points in the analysis of a cellular automaton is to know if the rule is capable of performing universal computation, or that it has an associated Turing machine. The rule 126 with memory, because it is logically universal and could simulate the type 3 and 2 languages of the Chomsky hierarchy, it indicates that the probability that the rule has an associated Turing machine is high, so it would be worth trying with some models.

3. **The rule allows us to simulate logic circuits:** This implication is born from the fact that it is logically universal, since with the NAND gate we can simulate any other gate by nesting several, we can receive an expression in boolean algebra and simulate it with this operation, we can also use the IF-THEN operation. to which they work in the same space and with the same equivalences.

In summary, the computational power of the rule is high and allowing us to simulate logic circuits opens us a step to make more complex composite structures.

Capítulo 7

SOL: Logical Operations Simulator

In this chapter we will describe this thesis product: a system which will be able to simulate every logic designs shown at the chapter 5. The content inside this chapter will show the basics for developing a complete system, which are: user's and sytem's requirements, system architecture, all modules within the architecture and the classes they will contain, the used technologies (programming languages, frameworks, etc.) and finally, the system's tests which will validate the requirement's fulfillments.

7.1. Requirements

In this section we will describe the essential requeriments for the system, in such a way SOL would not be able work without them (functional requirements), and the requirements that add value but are not imprescindible for this system to work (non functional requirements).

7.1.1. Functional requirements

1. **RF1. Save initial condition:** The system will allow the user to save the an evolution's initial condition, either in time t_0 or the first four times in the case of the rule with memory.

2. **RF2. Save the evolution:** The system will allow the user to save the automaton's evolution, in the stablished evolutions'quantity, either in plain text or

an export to PDF or PNG.

3. **RF3. Mosaics added per item:** The system will allow the user to fill with mosaics the evolution's blank spaces in order to use the rule with memory, either for the tiles for filtering or the particles.

4. **RF4. Multiple mosaics in the evolutions'space:** The system will allow the user to add multiple mosaics in the evolutions'space for its evaluation.

5. **RF5. Convertion of the first 4 evolutions to a regular matrix:** The system must transform the first four times of evolutions'space filled with mosaics in a regular matrix for evaluating with the memory rule.

6. **RF6. Initial data load:** The system will allow the user to set the system's initial condition for at least four different ways:

   - **Elemental way:** This means that user will load a live cell in the middle of the evolutions'space.
   - **Random way:** The user will be able to choose the zerosánd ones'percentages wanted to be in the initial condition.
   - **From a file:** The system will allow to load files with initial conditions to evaluate the automaton.
   - **Manually:** The user can enter the initial conditions manually.

7. **RF7. Tiles for filtering can be filtered:** The system will allow the user to filter the evolutions of the rule with memory applying a filter either for one or two tiles for filtering.

8. **RF8. Help option for user:** The system must show an option in menu to see the user manual.

9. **RF9. Logic operations load:** The system will allow the user to load the wanted logic operations.

7.1.2. Non functional requirements

1. **RNF1. The system is required to be handled instinctively:** This requirement means that user can handle the most of the system even if the user manual has not been read.

2. **RNF2. Ths system must be scalable:** Due to planned upgrades to the system in order to build a more complex one, this system's programming must be scalable to develop new features easier through code reusing.

7.2. System's architecture

Our system's architecture can be seen graphically in the figure 7.1. As we can see, there is our user, who is going to interact with the application once installed on their computer. Neither database instances nor servers were used due to the requirementsñeeds.

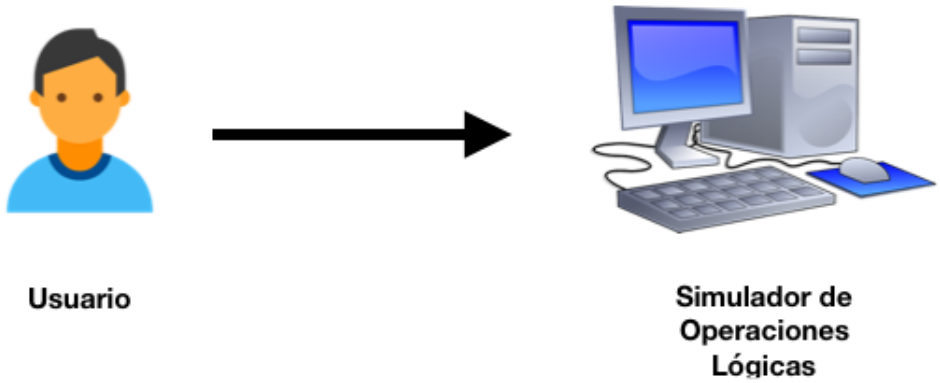


Figura 7.1: SOL's architecture

7.3. System's modules

Our system "Logical Operations Simulator", has 3 main modules as it can be seen in the figure 7.2, defined as:

1. **ACE:** Also known as the elemental cellular automaton module, inside it we will add the required classes to evaluate the 256 elemental rules.

2. **MemoryRule:** Also known as the elementary cellular automata's module, inside it we will have all the classes for mosaics handling in the rule, the interactions with them and the transformation to a regular matrix for its evaluation.

3. **LogicalOperations:** This module will allow us to use the defined logical operations to see them within the evolutions'space and understand thier behaviour.

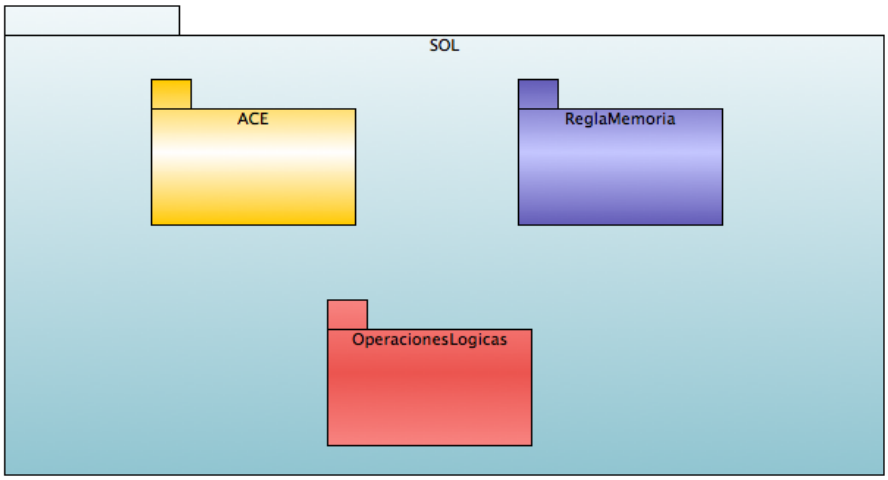


Figura 7.2: SOL's architecture

7.3.1. Elemental Cellular Automatons

In the figure 7.3 we can see the classes diagram of the three modules accoirding to model-view-controller. We have all our graphics interfaces, our controller and aour

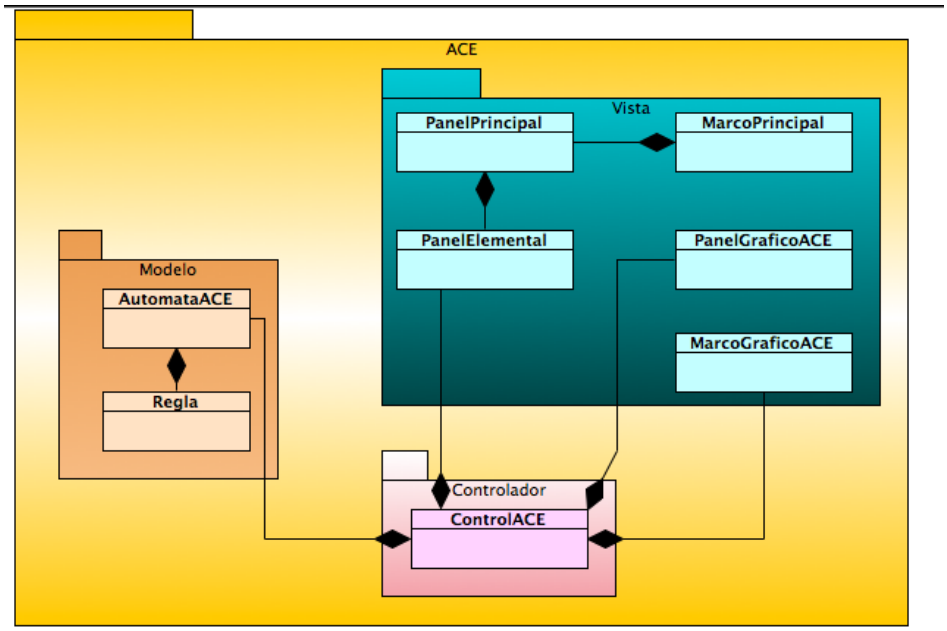


Figura 7.3: The classes which make up Elemental Cellular Automatons

model for elemental automatons. Now we will describe all classes per package, their attributes and methods and what they are for.

In the figure 7.4 we can see the classes'most important attributes and methods, as well as their interaction with the other modules through have them as attributes. Then we will describe in detail each module and class

For the model's package

In this package we have two classes: the Regla class, and the AutomataACE class, which consist in:

1. **Regla:** This class stores the definition of the rule we are going to plot and the neighbour interaction rules.

2. **AutomataACE:** Contains our rule, the evolutions'space and because of it will show us how it evolves in time, it will work with threads and updates the graph.

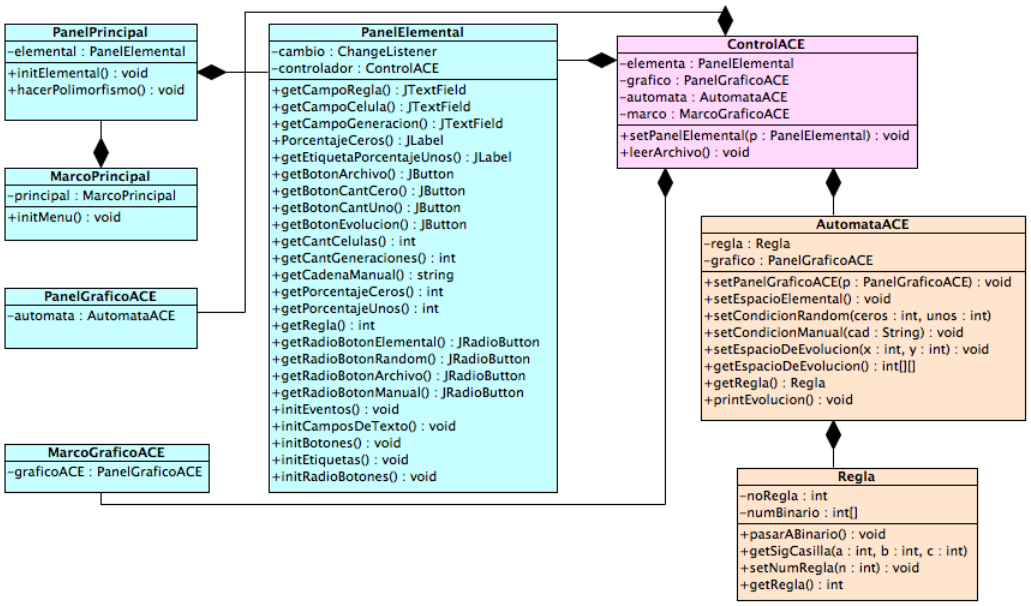


Figura 7.4: The classes which make up Elemental Cellular Automatons

For the view's package

In this package we have five classess: PanelPrincipal, PanelElemental, MarcoPrincipal, PanelGraficoACE and MarcoGraficoACE, which consist in:

1. **PanelPrincipal:** This class will be the panel which contain all other panels in the future. This will allow us to do polimorfism and the change between modules.

2. **PanelElemental:** This class contains all labels, radio buttons, buttons and text boxes so the user can introduce the data to evaluate the ACE's elemental rule.

3. **MarcoPrincipal:** This class is a JFrame which contains the PanelPrincipal and is only a container.

4. **PanelGraficoACE:** This class is the one which will plot the automaton's evolution in time, it will print the cell strips and show what happens in finite time intervals.

5. **MarcoGraficoACE:** This class is a JFrame too, it contains the PanelGraficoACE to show it to the user.

For the controller's package

In this package we have only one class: ControlACE, which will do all panel's events when evaluating an elemental rule is wanted, either the inputs or outputs pickings, and the plot through threads.

7.3.2. Rule 126 with memory

In the figure 7.5 we can see the classes diagram which the rule 126 with memory of 4 generations is composed of, we also can see that there are already several controllers for the rule, either for graphs or mosaics load and evaluation. Then we will describe each class, their functinality and purpose.

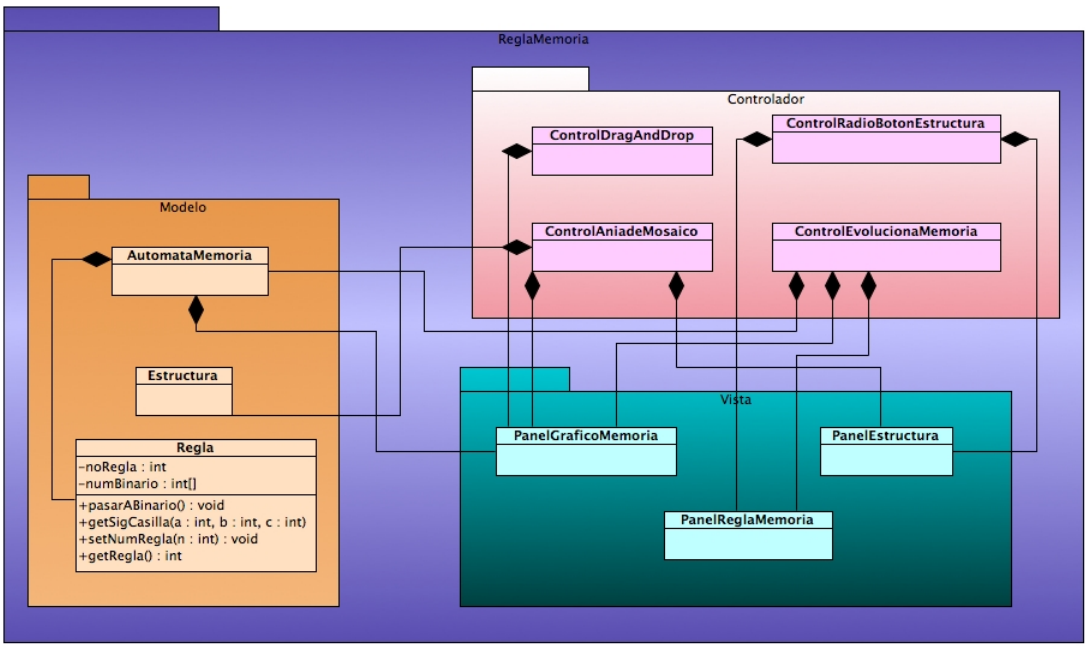


Figura 7.5: The classes which module of Rule 126 with memory is made of

In the figure 7.6 we can see the classes'most important attributes and methods, as well as their interaction with the other modules through have them as attributes. Then we will describe in detail each module and class.

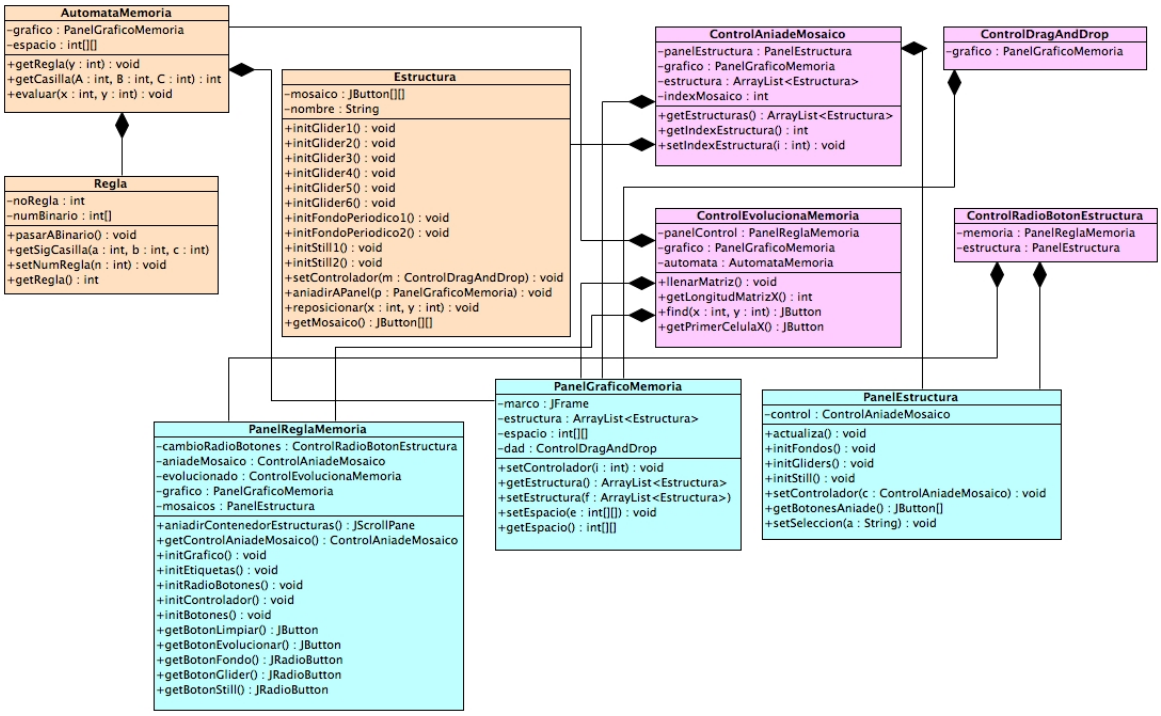


Figura 7.6: The described classes which compose the module of Rule 126 with memory

For the model's package

In this package we hace 3 classes: AutomataMemoria, Estructura and Regla, la Regla class is explained in the ACE section, that's why will only explain the two added classes:

1. **AutomataMemoria:** This automaton, unlike ACE's automaton, only works with the rule 126 with four generations'memory, it receives a matrix of four rows by any number of columns (cells), and calculates the generations to 50 (panel's size).

2. **Structure:** Now, in our system we will have 2 tiles for filtering, 2 still-life and 6 gliders, these 10 items will be treated as mosaics and will be added. but it has the same features: a group of cells which define them, that's why we defined them as Structures.

For the view's package

In this package we will found three classes: PanelGraficoMemoria, PanelEstructura and PanelReglaMemoria, let's describe them:

1. **PanelGraficoMemoria:** This panel will allow the user to drag the mosaics and set them freely, once clicked on the Evolve button, it will graph and show the regular matrix obtained from the mosaics.

2. **PanelReglaMemoria:** This panel will be our las control panel, but in this case it will be for the rule with memory, inside it, we will be able to see the settings with radio buttons and it will load the right panel to show and add the mosaics.

3. **PanelEstructura:** This panel will be inside the PanelReglaMemoria and show the right mosaics according to the user choice, as well as warn when the user clicks some add button according to the wanted selection to add it to PanelGraficoMemoria.

For the controller's package

This package has 4 controllers: ControlDragAndDrop, ControlAniadeMosaico, ControlRadioBotonEstructura and ControlEvolucionaMemoria, which have functions for either graphs or automaton's evolution, we describe them as:

1. **ControlDragAndDrop:** This controller will allow the user to drag the inserted mosaics in PanelGraficoMemoria, besides it limits the mosaic movements to every 10 pixels, which keeps a symmetry.

2. **ControlAniadeMosaico:** This controller is activated when in PanelEstructura the user clicks some add button, then it detects which mosaic is wanted to be added, and sends that mosaic's definition to PanelGraficoMemoria for its graphing.

3. **ControlRadioBotonEstructura:** This controller will make the selected radio button in PanelReglaMemoria, is the data to be loaded in PanelEstructura. If there is no change, it will refresh that panel.

4. **ControlEvolucionaMemoria:** This controller will take the mosaics from PanelGraficoMemoria and convert them to a regular matrix, after that it will update that panel sending that matrix and evolving to 50 generations.

7.3.3. Logical Operations

In the figure 7.7 we can see the classes diagram which composes our first logical operations'module. This implies that the three parts are: regular languages, context-free grammars, and logic gates. For those we will use several panels to show the options. Similarly we will use the model-view-controller.

In the figure 7.8 we can see the classes'most important attributes and methods, as well as their interaction with the other modules through have them as attributes. Then we will describe in detail each module and class:

For the model's package

For this package we have three classes: Filtro, AutomataMemoria and Regla. We have already described the classes AutomataMemoria and Regla in the previous

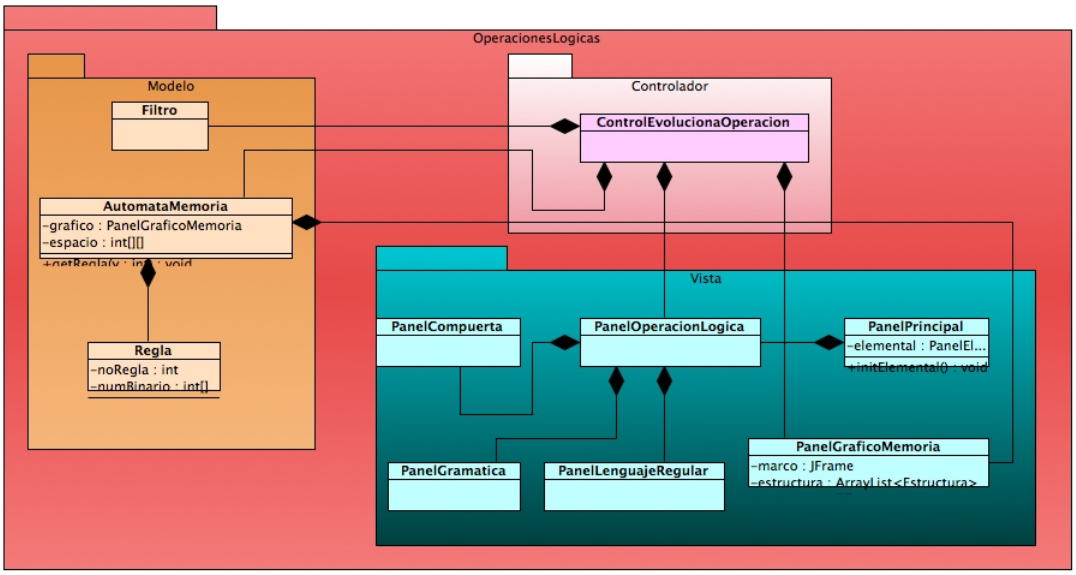


Figura 7.7: The clsses that compose the Loical Operations'module

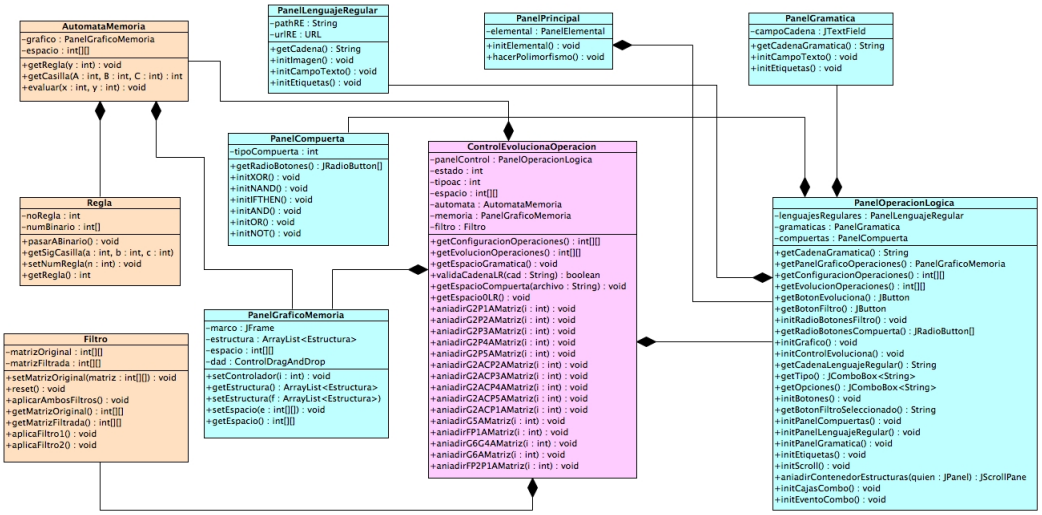


Figura 7.8: The described classes which compose the module of Rule 126 with memory

modules, so we will describe the class Filtro.

1. **Filtro:** This class will allow us to store the original matrix and filter it applying either the 1st tile for filtering, the 2nd tile for filtering or both at the same time.

For the view's package

For this one we have 6 classes, which will help us to generate the graphic interfaces that the user will interact with and will allow to choose the corresponding settings. This classes are enumerated and described below:

1. **PanelCompuerta:** This panel will allow us to see a table with the zerosánd onesávailable combinations to evaluate the chosen gate, as well as its equivalent equation in particles and some radio buttons for selecting the combination we want to see.

2. **PanelGramatica:** This panel will show us a text field to set the string we want to be generated with the production rules we have. Then we will see the string converted to particles collisions when the automaton's evolution is triggered by the user.

3. **PanelLenguajeRegular:** This panel will show us a text field to set the string we want to confirm to belong the language, to do this it will show us the auomaton that does this evaluation according to the chosen regular expression, evaluates the string, and prints the message telling whether it belongs or not.

4. **PanelOperacionLogica:** Thius panel will contain the previous three ones and will change to one or another according to user choices, as well as send the right information to our controller when evolving or filtering is wanted.

For the controller's package

We have only one class in this package, and its functionality is:

1. **ControlEvolucionaOperacion:** It will listen to our logical operations'panel, and when we click some evolve or filter button, it will listen to the information and send it to our automaton and graphic panel.

7.4. Technologies to use

In this section will be described the used technologies, as well as a short justification of why they were chosen. The programming languages, paradigms and versions used were:

1. **Object Oriented Programming:** We use this paradigm because it makes easier to see the rule as an object and automatons as another one.

2. **Java:** We use Java because it allows the compatibility between computers with different operating system, by installing the Java Virtual Machine, besides it is an object oriented language programming, which will allow us to use this paradigm. The JDK version used is 8.1.

7.5. System tests

In this section we will describe the made tests ti the system either for ideal conditions or possible errors when an user uses the application. The tests were madre by system's modules.

7.5.1. Elemental Cellular Automatons

On the system startup, it shows the elemental cellular automatons'module's screen and the option "Elemental" selected by default, if we set the rule rule 126 and the Elemental option keeps selected, in a space of 250 cells by 250 generations, we can see the result of the figure 7.9.

If we decide to choose the option "Random", some buttons will be enabled to change de zerosánd ones'percentages. In the figure 7.10 we can see 3 cases:

1. If we change the ones'percentages it will show a message where we can set the new value between 1 and 99 %

2. If we change the zeros'percentages it will show a message where we can set the new value between 1 and 99 %

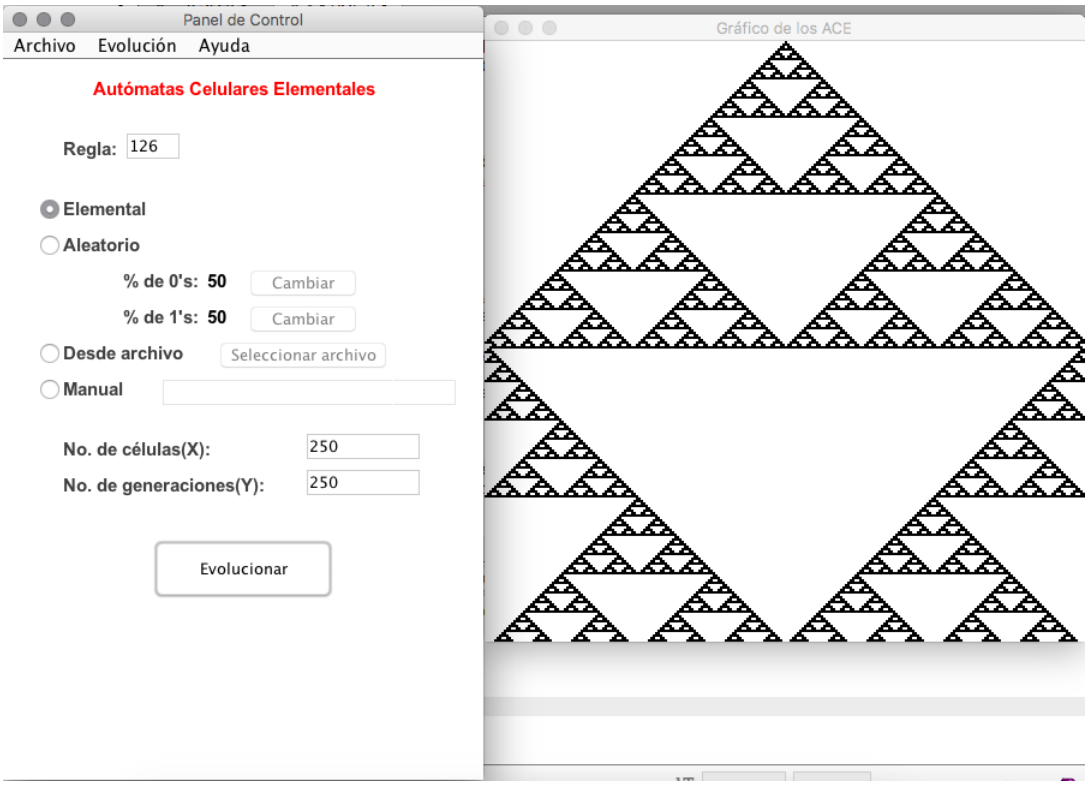


Figura 7.9: The system using the Elemental option

3. Let's suppose that we use 40 % of live cells and 60 % of dead cells, in a space of 200 cells by 250 generations,, we will see whats if shown in the figure 3.

If we choose the option "From file", the system will enable the button to upload the file, in the figure 7.11 we can see graphically the steps to test this option whicha are:

1. To illustrate it we will use the file of figure 1, which contains 120 characters of zeros and ones.

2. When clicking the button to upload a file, a window is deployed to choose it no matter which directory is in.

3. Once uploaded, the system will upload the cells value to 210 and it will allow us to graph it to the 250 generations we selected, for this example we used the rule 54.

The last option is to use it manually. In the figure 7.12 we can see how it would be handled manually, set the same string used in the File option, but using the rule 230 with 150 generations. We can see that the system shows the automaton evolving.

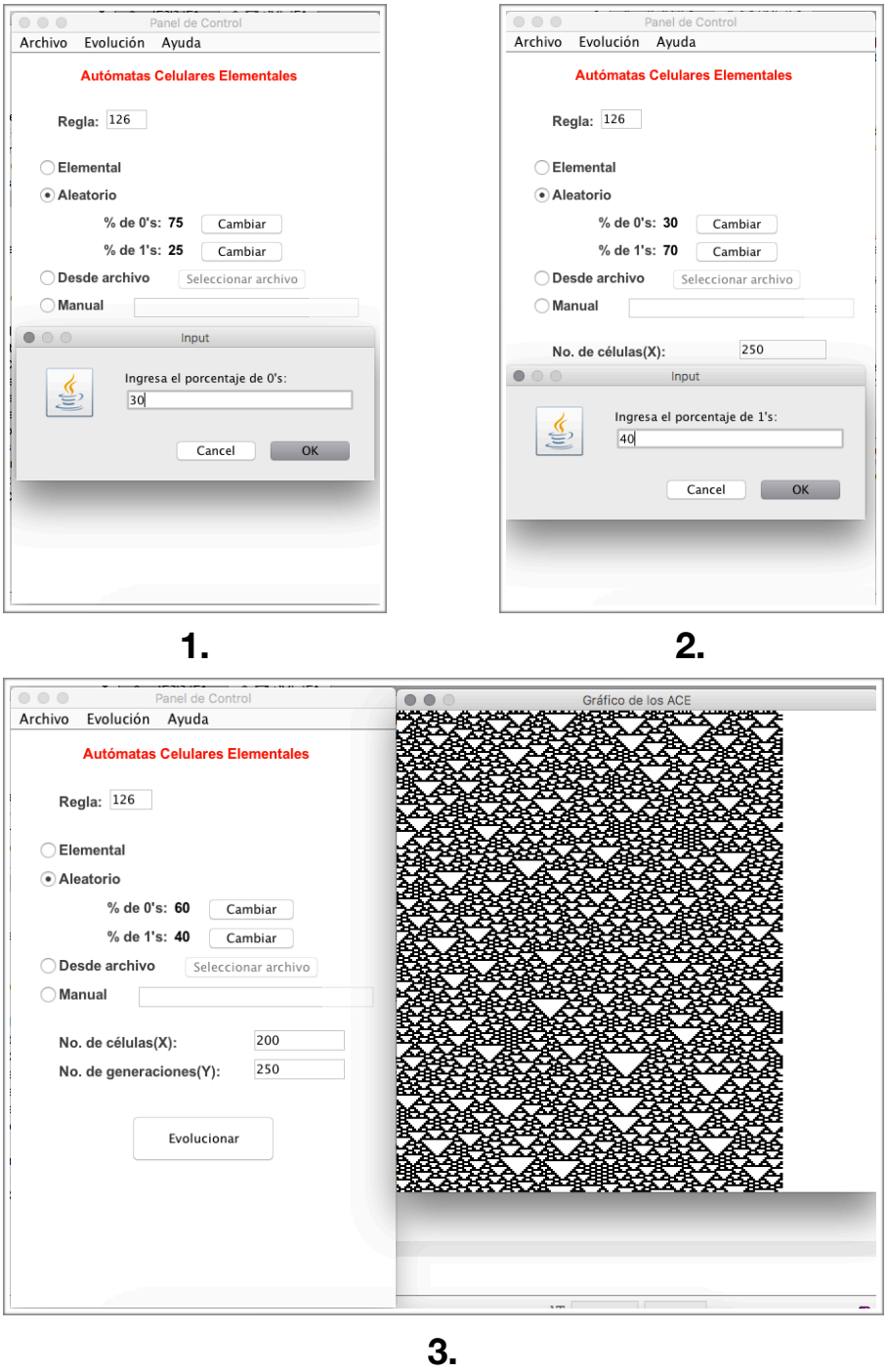


Figura 7.10: The system using the Random setting

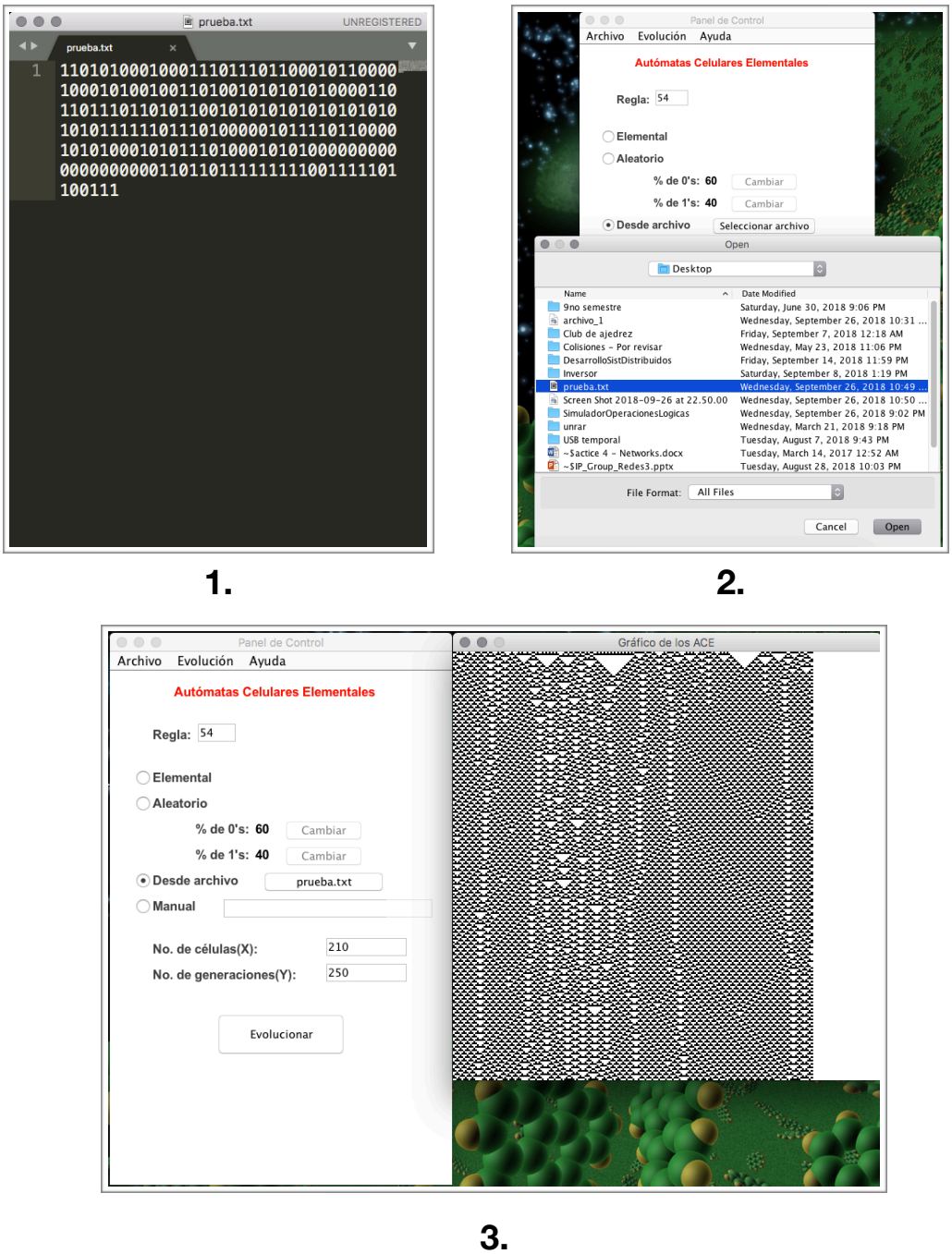


Figura 7.11: The system using the File setting

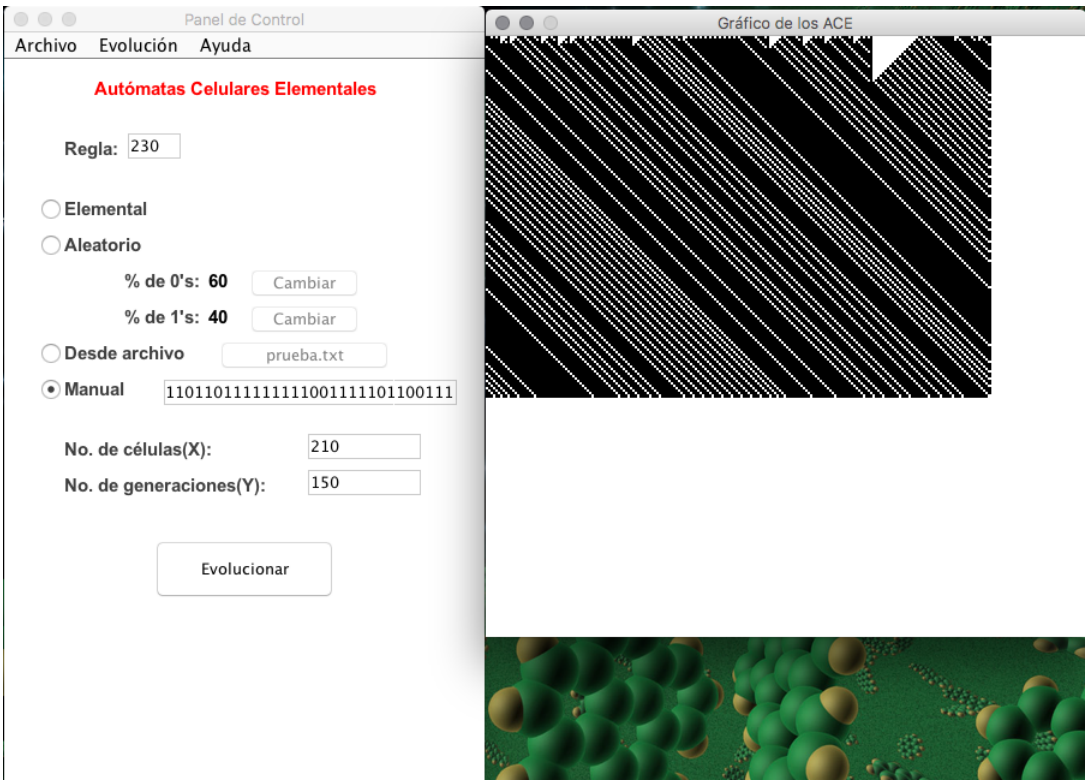


Figura 7.12: The system using the Manual option

7.5.2. Rule 126 with memory

Now we will describe this module by classifying the tests with tiles for filtering, gliders and still-life, to show how they are used and how are shown within the space.

Tiles for filtering

In the figure 7.13, we can see 4 pictures which represent the screens of module two's system, in the first one the startup system is shown, with the default tiles for filtering option, and shows a graphic panel where is pointed the first 4 lines with squares, those are our reference lines and will be the ones which will be taken to convert the mosaics. In the next one we see the added of a mosaic of tile for filtering 1 pointed marked with a blue square, we can add much more and move them to generate an evolution as it is shown in the third picture, the mosaics that are not in the first 4 rows will be ignored in evolution, finally in the last picture, we clicked to Evolve button, this will transform the array to a regular matrix and evolves it.

In the figure 7.14 we can see the case for the tile for filtering 2, again, we can add a tile for filtering block as it is shown in the first figure, after that, we can add much more and move them as it is shown in the second picture. The mosaics that are not in the first 4 lines will be ignored. Finally if we click the Evolve button, the result will be shown.

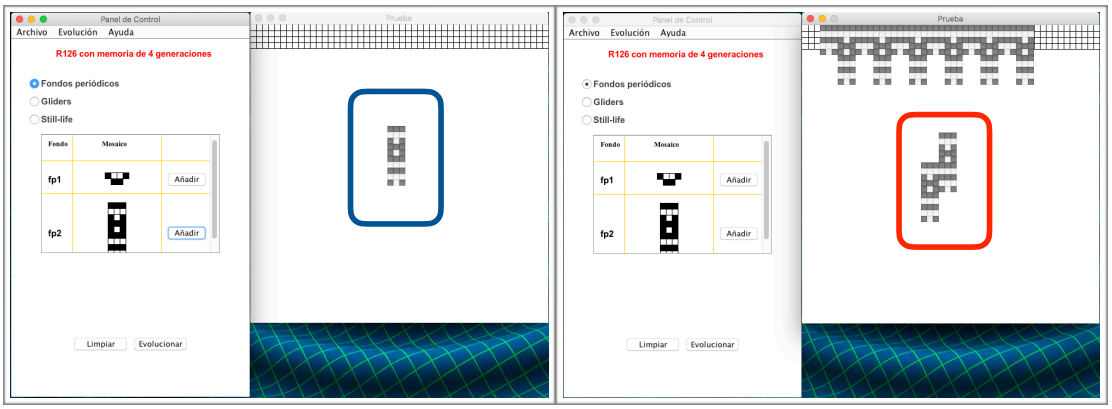
Gliders

In the figure 7.15 we can see how gliders work within these evolutions, we have 6 identified gliders, which coexist inside the tiles for filtering 1 and 2 for evolving, that's why we show the examples when we add and surround them with tiles for filtering for its evolving, again, the mosaics that are not in the first 4 lines will be ignored.

Still-life

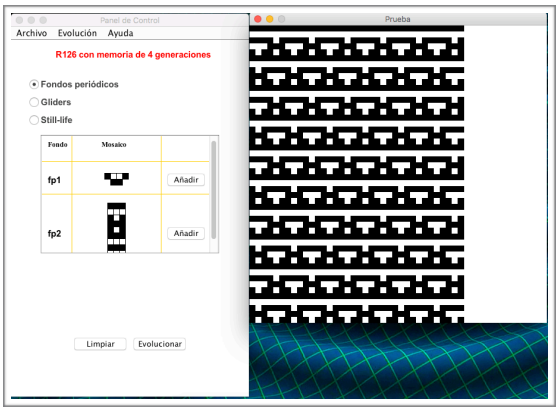
In the figure 7.16, we can see the cases of the two still-life identified with the corresponding tiles for filtering, as well as its evolution after becoming a regular

Figura 7.13: The evolution using the tile for filtering 1



Añadido de un único mosaico

Varios mosaicos, se ignoran los señalados con el cuadrado rojo



Evolución del arreglo

Figura 7.14: The evolution using the tile for filtering 2

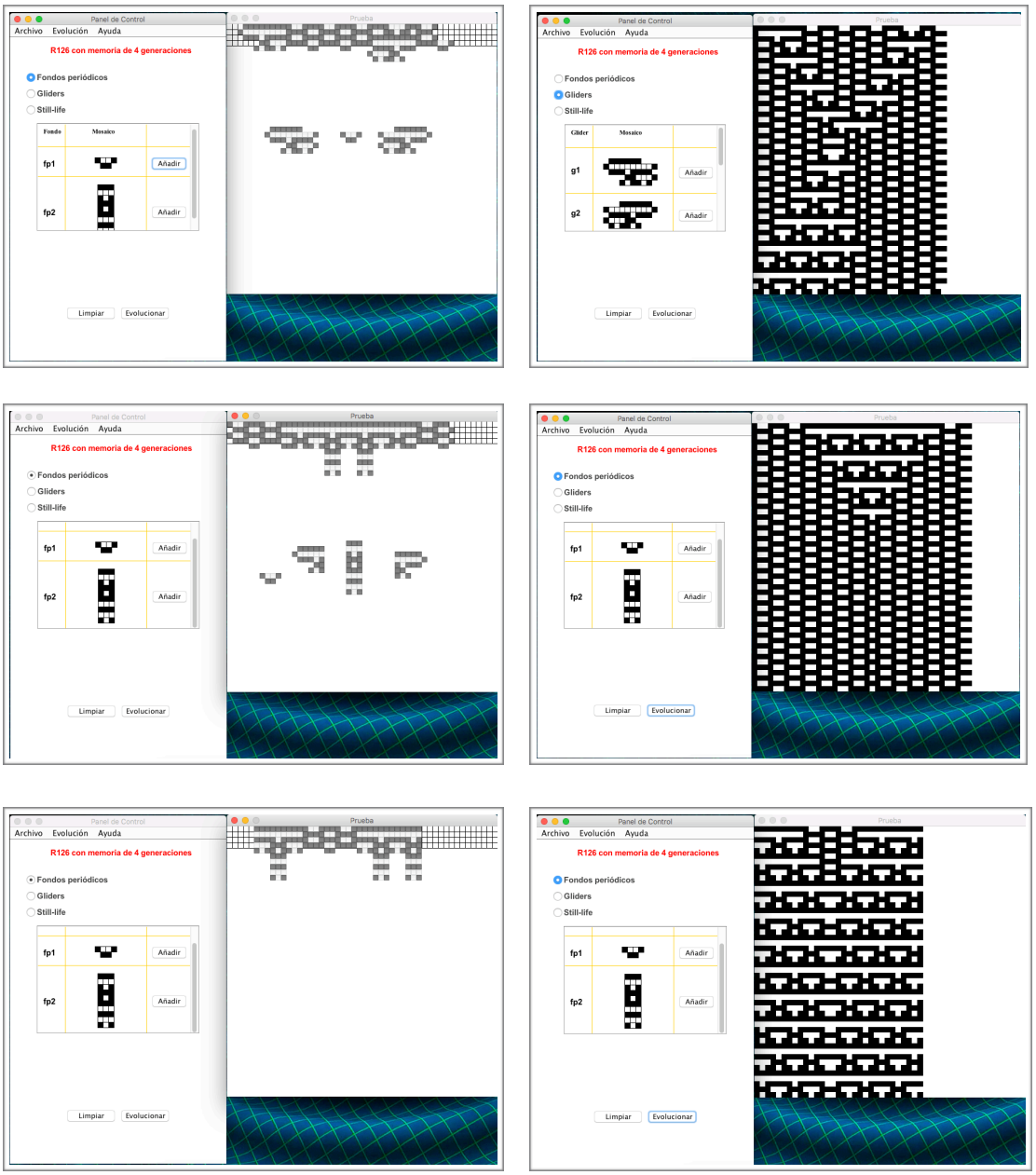


Figura 7.15: 6 gliders and their combinations with tiles for filtering

matrix and mosaics that define them, again, the mosaics that are not in the first 4 lines will be ignored.

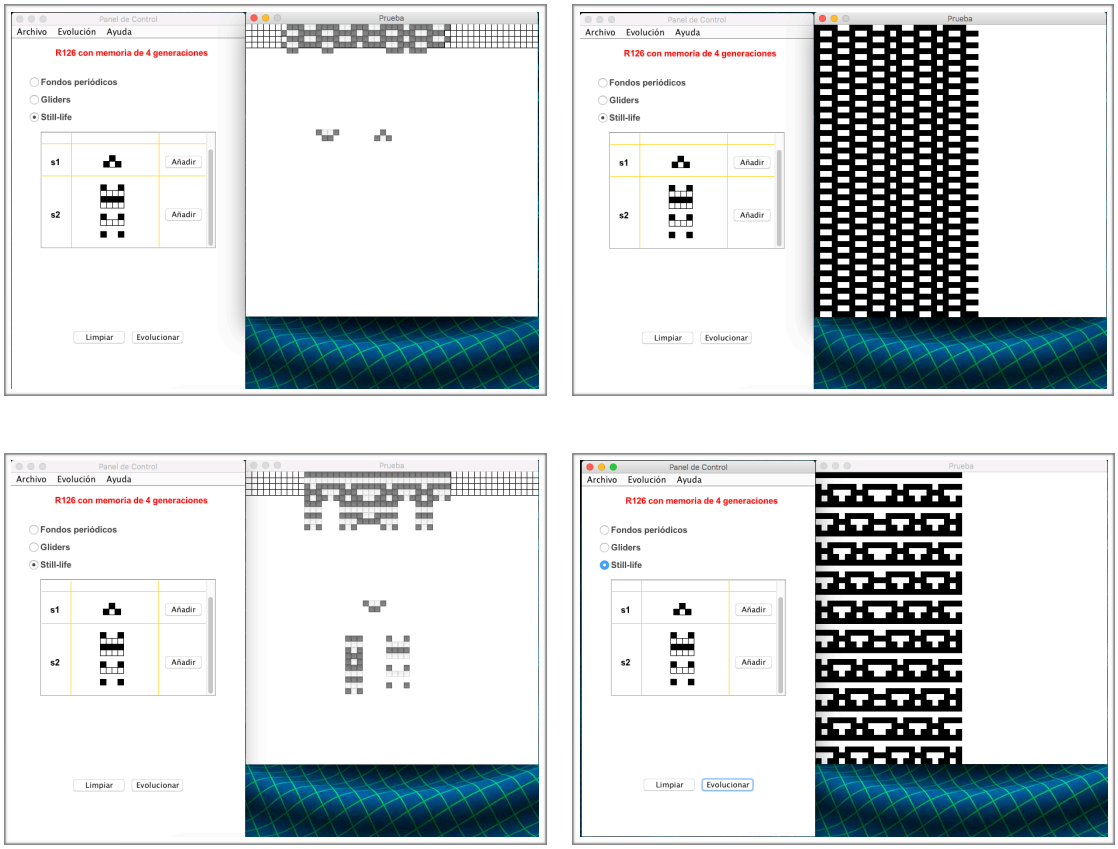


Figura 7.16: The 2 still-life and their combinatoins with tiles for filtering

7.5.3. Logical operations

In this section we will describe the tests of the our system's third and final module: the logical operations, we will describe available options and how the software works.

regular languages

In the figure 7.17 we can can see the demonstration of how settings of regular languages work. The first picture shows the module's startup, while the second one shows when we introduce an input string that does not belong to the language, it show a message telling that it does not belong and the equivalent evolution in particles. As well as we can also filter in three different ways: with the first tile for filtering, with the second tile for filtering, o applying both at the same time. In the figure 6 it is shown the message when we introduce a string which do belong.

Context-free grammars

In the figure 7.18 we can see how the Context-free grammar's option works, in which it is shown the production rules and again, it is generate the equivalent with particles. We can also filter and save: the configuration, that would save the first 4 evolutions of zeros and ones; and the evolution, that would save all the generated zeros and ones. We can export the graph with the evolutions to either PNG or PDF.

Logic gates

In the figure 7.19 we can see how it is used the logic gate option, we have 6 options: XOR, IF-THEN, NAND, AND, OR and NOT. The user can choose whichever option as it is shown in the figures and select a radio button of the desired combination of zeros and ones to be evoled. Once set, when clicking the Evolve button it will be shown the generated particles combination.

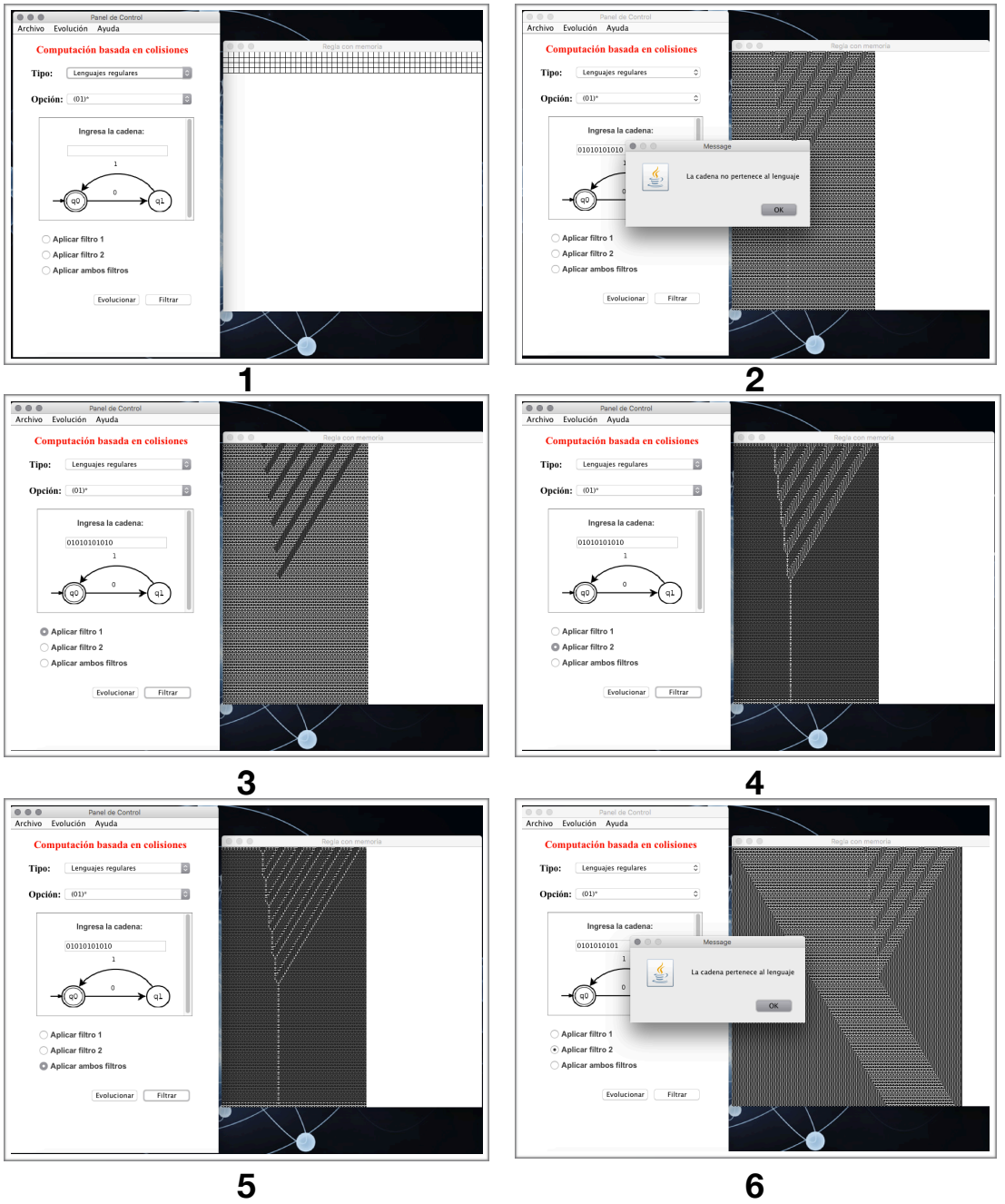


Figura 7.17: The regular expression's settings

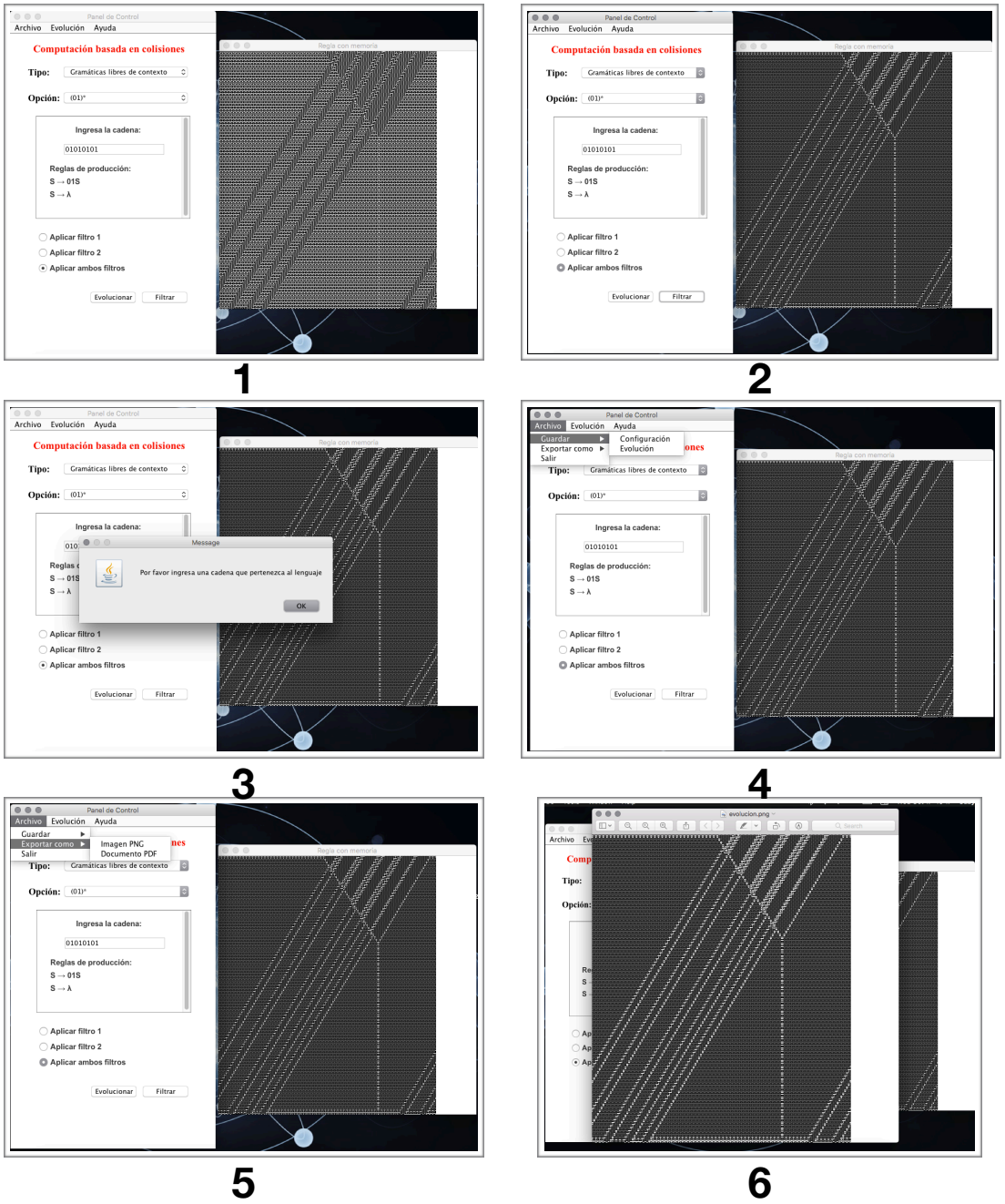


Figura 7.18: The settings to handle Context-free grammars

Figura 7.19: The settings to handle logic gates

Capítulo 8

Final results

This section describes the results obtained from this Thesis, as well as the conclusions and work that will be carried out in the future.

8.1. Celebration of late. Prof. Harold V. McIntosh Achievements 2017

This event was held on November 29 and 30, 2017 at the Faculty of Computer Science of the Autonomous University of Puebla, located in Av. San Claudio and 14 Sur, Ciudad Universitaria, C. P. 72570, Puebla, Mexico. The lectures were in memory of Dr. Harold V. McIntosh, pioneer of computing in Mexico, and which conducted research on the topics: REC, CONVERT, CAMEX, PLOT, Flexagons, Cellular Automata, complex variable, quantum mechanics, matrix theory , group theory and differential equations.

It involved renowned personalities within these areas such as: Leon Chua, Juan Carlos Seck Tuoh Mora, Kenichi Morita, Genaro Juarez Martinez, Andrew Adamatzky and Stephen Wolfram. My participation in the event was giving the conference "Recognizing chaos and complexity: case study in the elementary cellular automaton rule 126", which was held on November 29, 2017 from 6:00 pm to 6:20 pm. In this conference I presented the results of the study of the elementary rule (chapter 3), as well as some that had of the rule with memory (chapter 4). In the figure 8.1 we can observe some photographs, as well as the constancy obtained by

participating in said event.

Figura 8.1: Some photos of the event Celebration of Late Prof. Harold V. McIntosh Achievements

8.2. The Ninth International Conference on Complex Systems 2018

The ninth edition of the conferences on Complex Systems was held from July 22nd to July 27th, 2018 at the Hyatt Regency, located at 575 Memorial Drive, Cambridge, Massachusetts, USA 02139. During the event were held conferences, discussions tables and poster presentations on topics of non-linear dynamics, neural networks, mathematical systems, cellular systems, biomolecular systems and Artificial Intelligence. It involved world-renowned personalities in the ares of Complex Systems, such as: Stephen Wolfram, Yaneer Bar-Yam, H. Eugene Stanley, César Hidalgo, Spencer Wells, Albert-László Barabasi, Raúl Rojas and Steven Hassan.

Our participation consisted of giving a lecture entitled "Recognizing complex behavior emerging from chaos in cellular automata", which took place on July 23 from 15: 40-16: 00. In this conference the results of the elementary rule and the collisions of particles that generate objects within the rule were presented, as well as the gliders-gun that allow the existence of artificial life.

We also publish an article that has the same title of the conference in the book that contains the memories of the event, which is entitled "Unifying Themes in Complex Systems IX", Springer, pages. 82 - 90. In the figure 8.2 we can observe some photos of the event, as well as the published book that contains our article.

Figura 8.2: Some photos of the event The Ninth International Conference on Complex Systems

Capítulo 9

Conclusions

The cellular automata allow us to model behaviors that if we used another method to do it would be more complex, the study of these models, their behaviors, the capabilities they have to resemble reality, or in the case of this Terminal Work, to allow us to compute Something opens a way for us to process information and find results in a creative and efficient way.

In the academic part this TT allowed me to acquire study habits, supported me to develop the perseverance to learn something different, despite the failures and obstacles that arose when analyzing the model, continue believing that we could obtain interesting results from their study . I could understand in a broad way, I dare say even specialized, what is a cellular automaton, its characteristics, the importance of not only staying in what the definition says, of exploring further using a memory function, of recognizing patterns (a vital part of the TT and perhaps the one that cost me the most) and model them in a way that allows us to .exploit.our system and extract as much information as possible. I explored a branch of new knowledge, Complex Systems and non-conventional computing were pillars in the research, development and results presented in this document.

Opening up a bit the knowledge part, the potential that the memory showed by causing elements of complexity to emerge within the chaotic rules, which we can point out as rules that are not strongly chaotic, opens us an area of investigation not only with the automatons elementary cell phones, but with cellular automata that are considered chaotic. Now, these elements of complexity, as we already have the

history and all the study that the game of life has had, we can characterize them and study them in a similar way to what is known about this model, even using the languages, context-free grammars, etc, allows us to venture into the reversible and conservative computation, current issue and which has proved to have convenient characteristics.

In the personal field, I learned that constant work is the key to success, in the development of this TT one of the things that was presented very often was the uncertainty, being a research work did not exist the information that was required for to be able to perform the described software, it was required to carry out the study thoroughly and we did not know for sure if the rule was going to demonstrate a "docile" behavior, with which we could model the computations in an understandable, simple and effective way. I also understood that the search because the work is better than we could expect is a second key for an investigation to be exceptional, thanks to the demands and advice of my director of TT these results were obtained, as well as my personal demand for presenting a document in excellence.

I will conclude that researching in this field has a future, there are already many works dedicated to cellular automata, and we need many more to understand the computational universe, its limitations and the limitations that we can acquire when designing one way or another. Finally, the computational capabilities of a system can give us a strong support to solve problems that in one instance would take too much time.

Capítulo 10

Future Work

After all the analysis and the system made for this TT, the future work we can do is first that our system allows you to simulate any expression that is written with the boolean algebra using these logic gates. Also, because the rule already shows that it is logically universal, we know that the possibility of having an associated Turing machine is high, so we need to start a scan focused on merely finding it. Now, cellular automata have applications in many areas, we will focus these results on the understanding of cancer, the main motivation of this terminal work. One of the potentialities of the rule is the fact that we can combine periodic funds and the periodic fund that is mostly presented has a flexibility to be used very well, so we can take advantage of that flexibility to perform more complex computations, and even design a processor or higher things with collisions.

The idea that the software is updated so that anyone simply drag the gates that you want to use, nest them, and the software translates them into collisions, is a good idea to understand how the rule works, as well as the power they have to dock practically to any collision, and since after the exploration we know that the rule has symmetries with the collisions, we can use these symmetries as supports for the coupling.

There is work to be done with these results, as well as with things that were not yet clear from the rule, but because of the current results, we know that they are explorations and investigations that need to be done since the rule has shown potential and very high possibilities of to be able to simulate bigger things.

Bibliografía

[1] A. Adamatzky. *Identification of Cellular Automata*. Taylor y Francis, 1 ed$^{\underline{\text{ón}}}$., 1994.

[2] A. Adamatzky. *Computing in Nonlinear Media and Automata Collectives*. Institute of Physics Publishing, 1 ed$^{\underline{\text{ón}}}$., 2001.

[3] A. Adamatzky. *Collision-Based Computing*. Springer, 1 ed$^{\underline{\text{ón}}}$., 2002.

[4] A. Adamatzky. *Advances in Unconvetional Computing. Volume 1: Theory*. Springer, 1 ed$^{\underline{\text{ón}}}$., 2017.

[5] R. Alonso-Sanz y M. Martín. Elementary cellular automata with memory. *Complex Systems*, 14:99–126, 2003.

[6] Y. Bar-Yam. *Dynamics of Complex Systems*. Addison Wesley, 1 ed$^{\underline{\text{ón}}}$., 1997.

[7] E. R. Berlekamp, J. H. Conway, y R. L. Guy. *Information and Transmission in Cellular Automata*. Winning Ways for your Mathematical Plays vol. 2: Games in Particular, 2 ed$^{\underline{\text{ón}}}$., 1982.

[8] E. Borriello y S. I. Walker. An information-based classfication of elementary cellular automata. págs. 1–8, 2017.

[9] M. Cook. Universality in elementary cellular automata. *Complex Systems*, 15:1–40, 2004.

[10] J. C. Seck-Tuoh-Mora G. J. Martínez, A. Adamatzky y R. Alonso-Sanz. How to make dull cellular automata complex by adding memory: Rule 126 case study. *Complexity*, 15(6):34–49, 2010.

[11] D. A. R. Gomez. Descripción y aplicaciones de los autómatas celulares. *Verano de Investigación 2011*, págs. 3–5, 2011.

[12] A. C. González. La medida del caos en autómatas celulares que presentan comportamiento colectivo no trivial. *Computer Based Learning Unit, University of Leeds*, págs. 3–7, 1998.

[13] P. Grogono. Dynamic systems. *Stanislaw Lem*, págs. 8–12, 2005.

[14] J. E. Hopcroft, R. Motwani, y J. D. Ullman. *introducción a la Teoría de autómatas, lenguajes y computación*. Addison Wesley, 1 ed$^{\underline{\text{ón}}}$., 2007.

[15] K. Culik II y S. Yu. Undecidability of ca classification schemes. *Complex Systems*, 2:177–190, 1985.

[16] R. Klages. Introduction to dynamic systems. *Queen Mary, University of London*, págs. 12–19, 2008.

[17] O. Martin, A. M. Odlyzko, y S. Wolfram. Algebraic properties of cellular automata. *Communications in Mathematical Physics*, 93:219–258, 1984.

[18] G. J. Martínez. Grados de reversibilidad en autómatas celulares lineales. págs. 1–31, 1998.

[19] G. J. Martínez. A note on elementary cellular automata classification. págs. 1–25, 2013.

[20] G. J. Martínez y A. Adamatzky. Cellular automaton super-collider: An abstract model. *MIT Technology Review*.

[21] G. J. Martínez, A. Adamatzky, y R. Alonso-Sanz. Descripción y aplicaciones de los autómatas celulares. págs. 10–39, 2013.

[22] G. J. Martínez, A. Adamatzky, F. Chen, y L. Chua. On soliton collisions between localizations in complex elementary cellular automata: Rules 54 and 110 and beyond. *Complex Systems*, 21:1–26, 2012.

[23] G. J. Martínez, A. Adamatzky, y H. V. McIntosh. Complete characterization of structure of rule 54. *Complex Systems*, 23:1–33, 2014.

[24] G. J. Martínez, A. Adamatzky., y H. V. McIntosh. A computation in a cellular automaton collider rule 110. págs. 1–37, 2016.

[25] G. J. Martínez, A. Adamatzky, y K. Morita. Logical gates via gliders collisions. *Journal of Cellular Automata*, 13(4):325–346, 2018.

[26] G. J. Martínez, A. Adamatzky, y C. R. Stephens. Cellular automaton supercolliders. págs. 1–24, 2011.

[27] G. J. Martínez, H. V. McIntosh, y J. C. Seck-Tuoh-Mora. Gliders in rule 110. *International Journal of Unconvetional Computing*, 2:1–49, 2005.

[28] G. J. Martínez, H. V. McIntosh, J. C. Seck-Tuoh-Mora, y S. Vergara. Rule 110 objects and other collision-based constructions. págs. 1–25, 2006.

[29] G. J. Martínez, H. V. McIntosh, J. C. Seck-Tuoh-Mora, y S. V. C. Vergara. Determining a regular language by glider-based structures called phases fi_1 in rule 110. págs. 1–26, 2006.

[30] G. J. Martínez, J. C. Seck-Tuoh-Mora, y H. Zenil. Wolfram's classification and computation in cellular automata classes iii and iv. págs. 1–27, 2012.

[31] H. V. McIntosh. Linear cellular automata. págs. 1–26, 1987.

[32] A. J. Morales, C. Gershenson, D. Braha, A. A. Minai, y Y. Bar-Yam. *Unifying Themes in Complex Systems IX*. Springer, 1 ed$^{\underline{\text{ón}}}$., 2018.

[33] J. V. Neumann. *Theory of Self-Reproducing Automata*. Library of Congress, 1 ed$^{\underline{\text{ón}}}$., 1966.

[34] N. H. Packard y S. Wolfram. Two-dimensional cellular automata. *Journal of Statistical Physics*, 38(5), 1985.

[35] T. Panitanarak. Cellular automata. *CS6800 Summer*, págs. 2–4, 2009.

[36] J. C. Seck-Tuoh-Mora. Autómatas celulares lineales reversibles. págs. 5–21, 1997.

[37] M. A. Shereshevsky. Lyapunov exponents for one-dimensional cellular automata. *Journal of Nonlinear Science*, 2:1–8, 1992.

[38] S. Wolfram. Cellular automata as models of complexity. *CALT*, 66(938):1–12, 1982.

[39] S. Wolfram. Cellular automata. *Los Alamos Science*, págs. 4–9, 1983.

[40] S. Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55(3):600–644, 1983.

[41] S. Wolfram. Cellular automata as models of complexity. *Nature*, 311(5985):1–7, 1984.

[42] S. Wolfram. Computation theory of cellular automata. *Communications in Mathematical Physics*, 96:15–57, 1984.

[43] S. Wolfram. Universality and complexity in cellular automata. *Physica*, 10:1–35, 1984.

[44] S. Wolfram. Cellular automaton supercomputing. págs. 1–9, 1986.

[45] S. Wolfram. Tables of cellular automaton properties. *Theory and Applications of Cellular Automata*, págs. 485–557, 1986.

[46] S. Wolfram. *A New Kind of Science*. Wolfram Media, 1 ed$^{\underline{\text{ón}}}$., 2002.

[47] A. Wuensche y M. Lesser. *The Global Dynamics of CellularAutomata*. Addison-Wesley, 1 ed$^{\underline{\text{ón}}}$., 1992.

Glossary

Attractor It is the region of the state space in which there are no exit routes. 24

Binary collisions They are collisions that only involve two particles within the evolution space. 80

Cellular Automata It is an algebraic structure composed of states, dimension, alphabet and rules of evolution.. 28

Cellular Automata with memory They are cellular automata that evaluate more than one evolution over time and have a memory function. 30

Collision points They are evolutions where a glider or a stationary structure collide with each other in some evolution and generate a new pattern. 72

Collision-based computing It is a model that allows us to perform operations similar to those of a computer using elements that when interacting between them generate the bases to compute something. 147

Complex Systems They are systems that have characteristics such as emerging, non-trivial collective behavior and being non-linear.. 46

De Bruijn Diagrams The paths in a de Bruijn diagram can represent chains, configurations, or classes of configurations in the evolutions space. 42

Dynamic System It is a system that, in reality or conceptually, evolves over time. 23

Elementary Cellular Automata They are Cellular Automata that work in one dimension and their evolution rules only work with their immediate neighbors. 35

Glider This type of structure is characterized by being a set of cells that seem to move either to the left or to the right and maintain their shape. 61

Glider-gun They are a set of cells that seem to "shoot" gliders, we can also know them as generators of gliders. 61

Linearity It replicates how "stable" a system is. 25

Lyapunov Exponents They are a set of numbers that are usually used to detect the presence of chaos in dynamic systems. 27

Still-life They are similar to gliders, except that with the difference that they do not move to the left or to the right, they remain in the same position.. 61

Ternary collisions They are collisions that only involve three particles within the evolution space. 93

Tile for filtering They are a set of cells that look like a mosaic that repeats infinitely, but allow the existence of gliders, gliders-gun and stationary structures. 61

Apéndice A

Attractors of the rule 126

In this section we will place all the descriptive tables of the attractors (mass, period, number of leaves and grade) for the attractors of length 2 to length equal to 15 because this is not found in the work done by Andrew Wuensche [47], from the length 16 to 24 the table and some images of the most interesting attractors will be placed. The images shown can be understood by following the following color code:

1. A line of cherry color indicates an edge.
2. A blue dot indicates a vertex.

A.1. Attractors with length equal to 2

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**
1	4	1	2	3

Cuadro A.1: Attractors for strings of length equal to 2

A.2. Attractors with length equal to 3

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**
1	8	1	6	3

Cuadro A.2: Attractors for strings of length equal to 3

A.3. Attractors with length equal to 4

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**
1	8	1	6	3
2	4	2	2	2
3	4	2	2	2

Cuadro A.3: Attractors for strings of length equal to 4

A.4. Attractors with length equal to 5

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**
1	12	1	10	3
2	4	2	2	2
3	4	2	2	2
4	4	2	2	2
5	4	2	2	2
6	4	2	2	2

Cuadro A.4: Attractors for strings of length equal to 5

A.5. Attractors with length equal to 6

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**
1	52	1	35	6
2	4	2	2	2
3	4	2	2	2
4	4	2	2	2

Cuadro A.5: Attractors for strings of length equal to 6

A.6. Attractors with length equal to 7

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**
1	128	1	91	6

Cuadro A.6: Attractors for strings of length equal to 7

A.7. Attractors with length equal to 8

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**
1	136	1	106	7
2,3,4,5	28	6	20	3
6,7	4	2	2	2

Cuadro A.7: Attractors for strings of length equal to 8

A.8. Attractors with length equal to 9

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**
1	80	1	78	3
2 al 9	44	6	33	4
10, 12 al 19	4	2	2	2
11	44	6	33	4

Cuadro A.8: Attractors for strings of length equal to 9

A.9. Attractors with length equal to 10

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**
1	124	1	122	3
2,3,5,8,11	104	4	82	7
4,7,10,13,17	56	6	44	4
6,9,12,14,16	12	2	7	4
15,18 al 26	4	2	2	2

Cuadro A.9: Attractors for strings of length equal to 10

A.10. Attractors with length equal to 11

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**
1	200	1	198	3
2 al 5, 7, 8, 11, 12, 15, 16, 20	136	4	109	7
6, 9, 10, 13, 14, 17, 19, 22 al 25	24	2	18	5

Continued on the next page

Cuadro A.10 – *Continued from previous page*

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**
18, 21	44	11	33	2

Cuadro A.10: Attractors for strings of length equal to 11

A.11. Attractors with length equal to 12

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**
1	548	1	495	9
2, 3, 5, 7, 10, 15	284	14	242	6
4, 6, 9, 13, 17, 18	188	4	154	7
8, 11, 16	124	2	96	7
12, 14	168	36	120	3
19, 20	4	2	2	2

Cuadro A.11: Attractors for strings of length equal to 12

A.12. Attractors with length equal to 13

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**
1	548	1	495	9
1	2160	1	1820	10
2 al 9, 11, 13 al 16	428	14	371	6
10, 12	208	39	143	3
17 al 29	4	2	2	2

Continued on the next page

Cuadro A.12 – *Continued from previous page*

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**

Cuadro A.12: Attractors for strings of length equal to 13

A.13. Attractors with length equal to 14

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**
1	11932	1	10327	19
2 al 5, 7, 9, 10	588	14	520	8
6, 8	84	14	56	3
11, 12, 16, 17, 20, 23, 26, 27, 31, 32, 35, 38, 41, 42	6	2	4	2
13, 14, 15, 18, 19, 21, 22, 24, 25, 28, 29, 30, 33, 34, 36, 37, 39, 40, 43, 44, 45	4	2	2	2

Cuadro A.13: Attractors for strings of length equal to 14

A.14. Attractors with length equal to 15

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**
1	32448	1	28621	21
2, 12, 22, 32, 42	10	2	8	2
3 al 11, 13 al 21, 23 al 31, 33 al 41, 43 al 51	6	2	4	2

Continued on the next page

Cuadro A.14 – *Continued from previous page*

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**

Cuadro A.14: Attractors for strings of length equal to 15

A.15. Attractors with length equal to 16

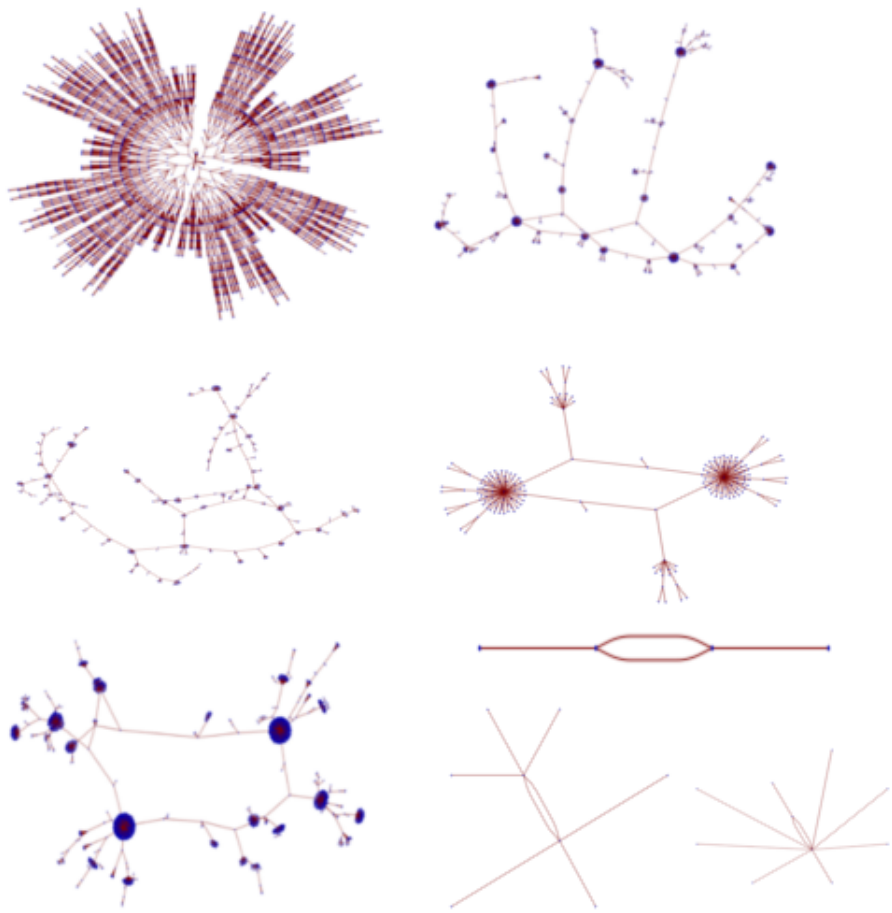


Figura A.1: Attractors with length equal to 16

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**
1	31560	1	28290	22
2, 3, 4, 6, 8, 12, 16, 22	1968	14	1864	8
5, 7, 9, 13, 18, 25, 29, 34	904	14	784	13
10, 11, 14, 15, 17, 19, 20, 21, 23, 24, 27, 28, 30, 31, 33, 35	620	14	510	14

Continued on the next page

Cuadro A.15 – *Continued from previous page*

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**
26, 32, 36, 37	180	6	160	4
38, 40, 42, 47, 48, 51, 54, 55, 58, 63, 64, 67, 70, 71, 74, 79	10	2	8	2
39, 41, 43 al 46, 49, 50, 53, 56, 57, 59 al 62, 65, 66, 69, 72, 73, 75 al 78	8	2	6	2
52, 68	4	2	2	2

Cuadro A.15: Attractors for strings of length equal to 16

A.16. Attractors with length equal to 17

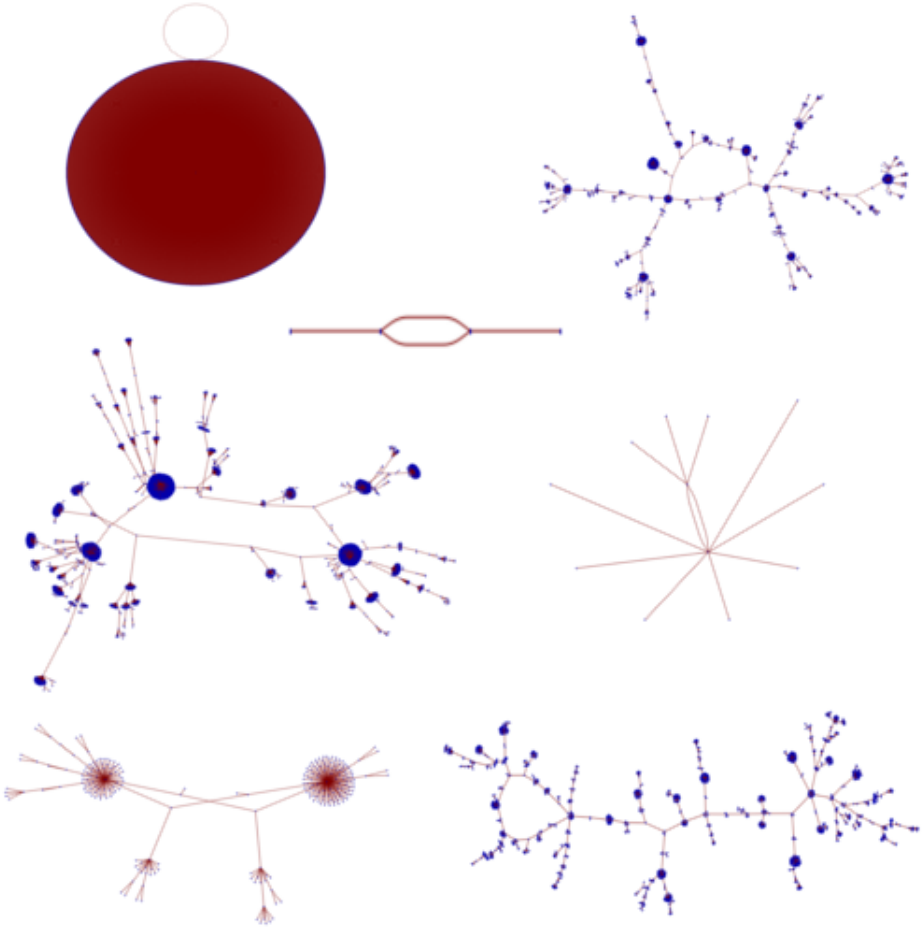


Figura A.2: Attractors with length equal to 17

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**
1	3572	1	3570	3
2 al 5, 7, 8, 11, 12, 16, 18, 24, 26, 35, 38, 46, 50, 60	2988	14	2806	9

Continued on the next page

Cuadro A.16 – *Continued from previous page*

No. of attractor	**Mass**	**Period**	**Num. of leafs**	**Grade**
6, 9, 10, 13, 14, 19, 21, 29, 31, 39, 40, 42, 48, 52, 59, 63, 64	1696	14	1511	13
15, 17, 20, 22, 23, 25, 27, 28, 30, 32, 33, 34, 36, 37, 41, 43, 44, 45, 47, 49, 53 al 58, 61, 62, 68 al 72	1284	14	1115	23
51, 65, 66, 67, 73, 75 al 86	232	6	211	4
87, 89 al 92, 94, 101 al 105, 108, 115 al 119	12	2 10	2	
88, 93, 95 al 100, 106, 107, 109 al 114, 120	4	2	2	2

Cuadro A.16: Attractors for strings of length equal to 17

A.17. Attractors with length equal to 18

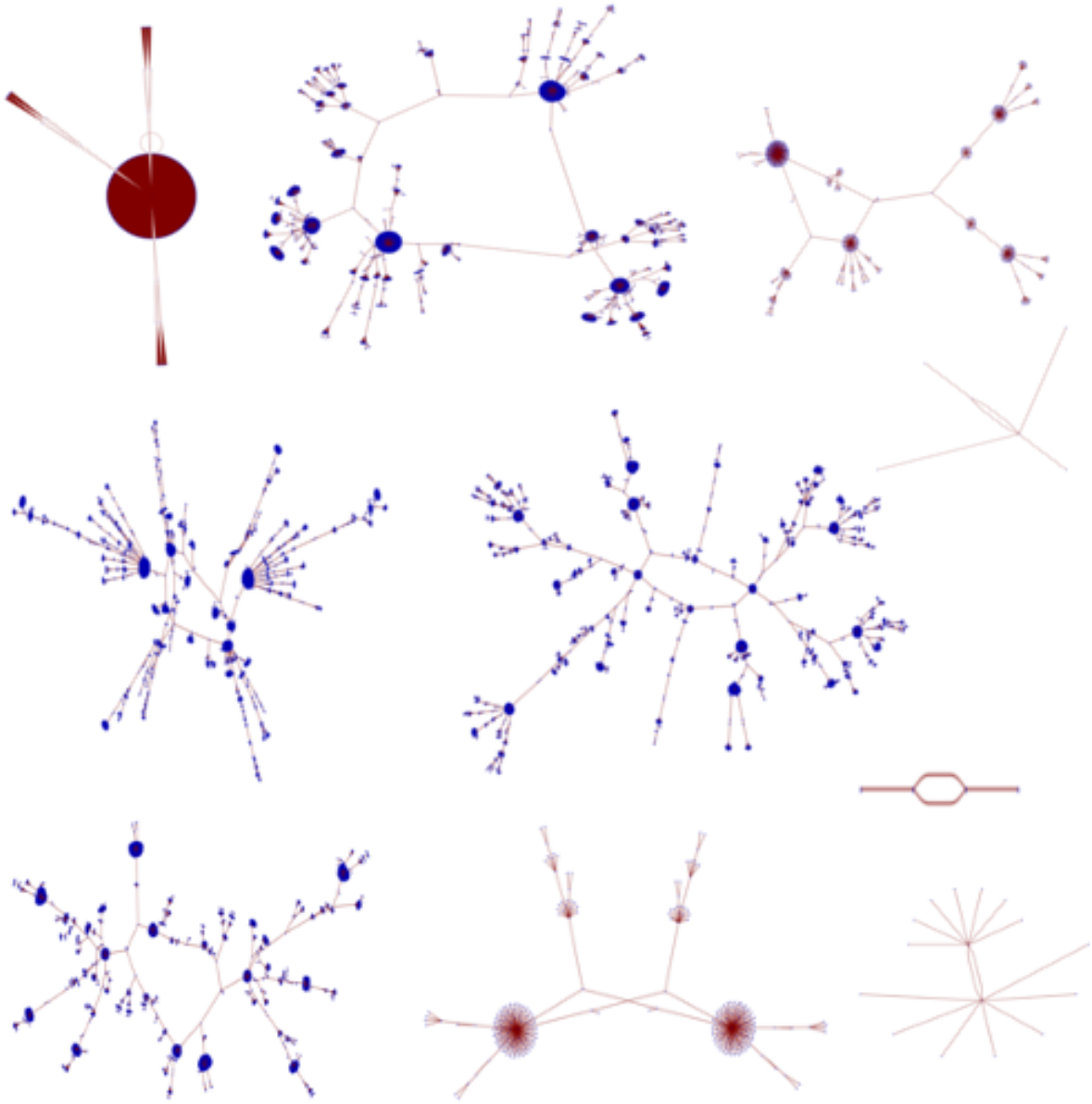


Figura A.3: Attractors with length equal to 18

A.18. Attractors with length equal to 19

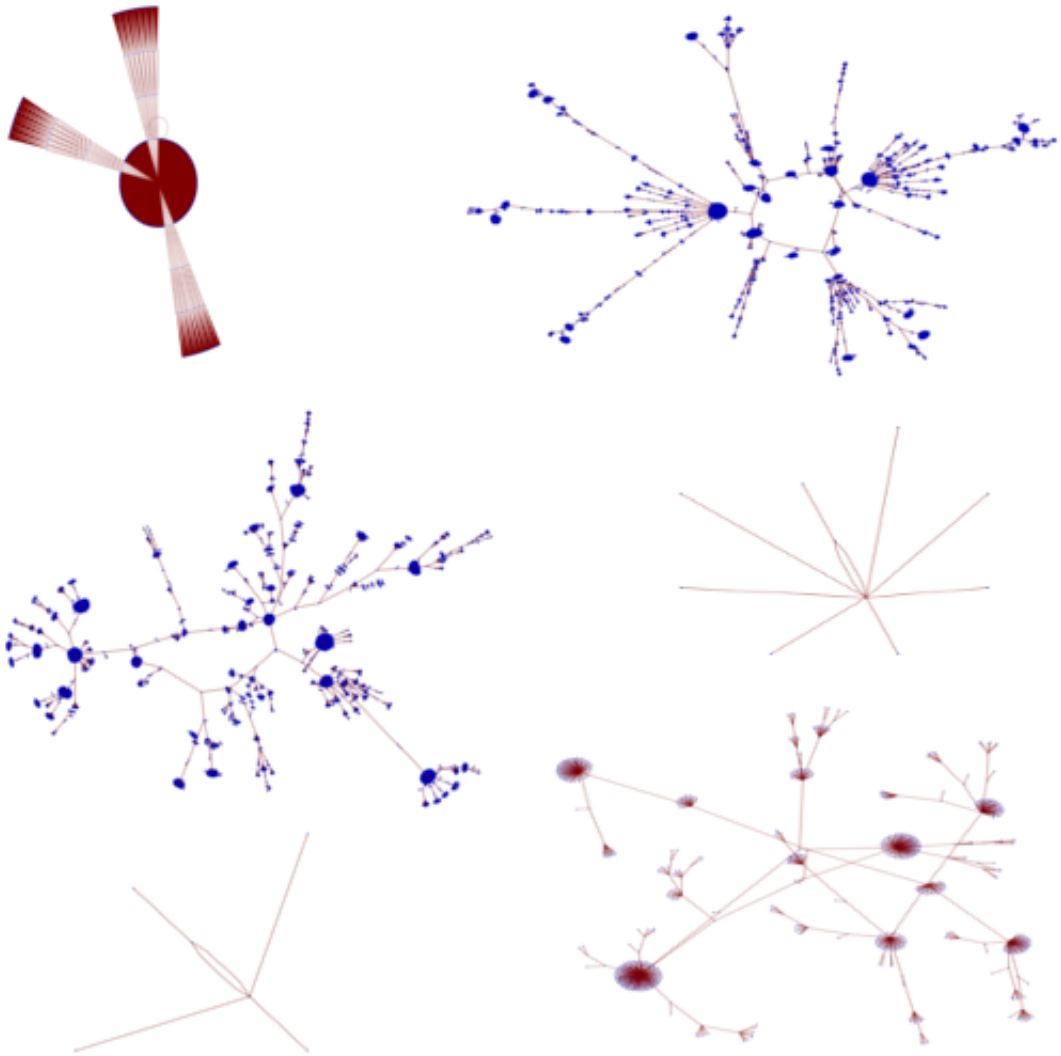


Figura A.4: Attractors with length equal to 19

A.19. Attractors with length equal to 20

The following attractors show us a chaotic behavior and in different ways, some of them have extensive gardens of Eden, while others have a large ring but small gardens, and others only have a large number of leaves. They can be observed in detail in the following figure:

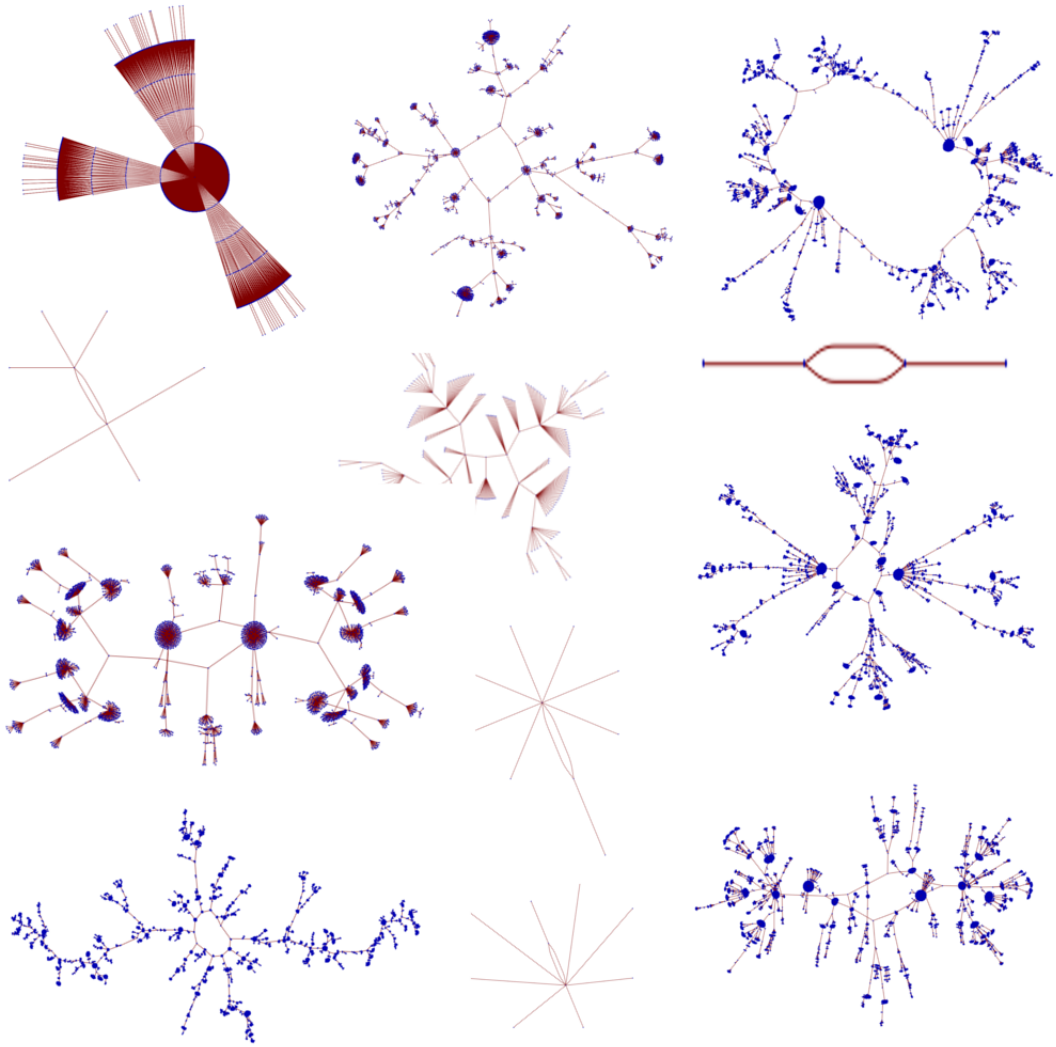


Figura A.5: Attractors with length equal to 20

A.20. Attractors with length equal to 21

The same attractor, however the amount of Garden of Eden to reach it has increased considerably. The attractors show chains that generate a pattern that will repeat n times in certain steps of time, however, it may be that some attractors contain others, as in this case this attractor to own more Garden of Eden than the previous one we can conclude which contains it:

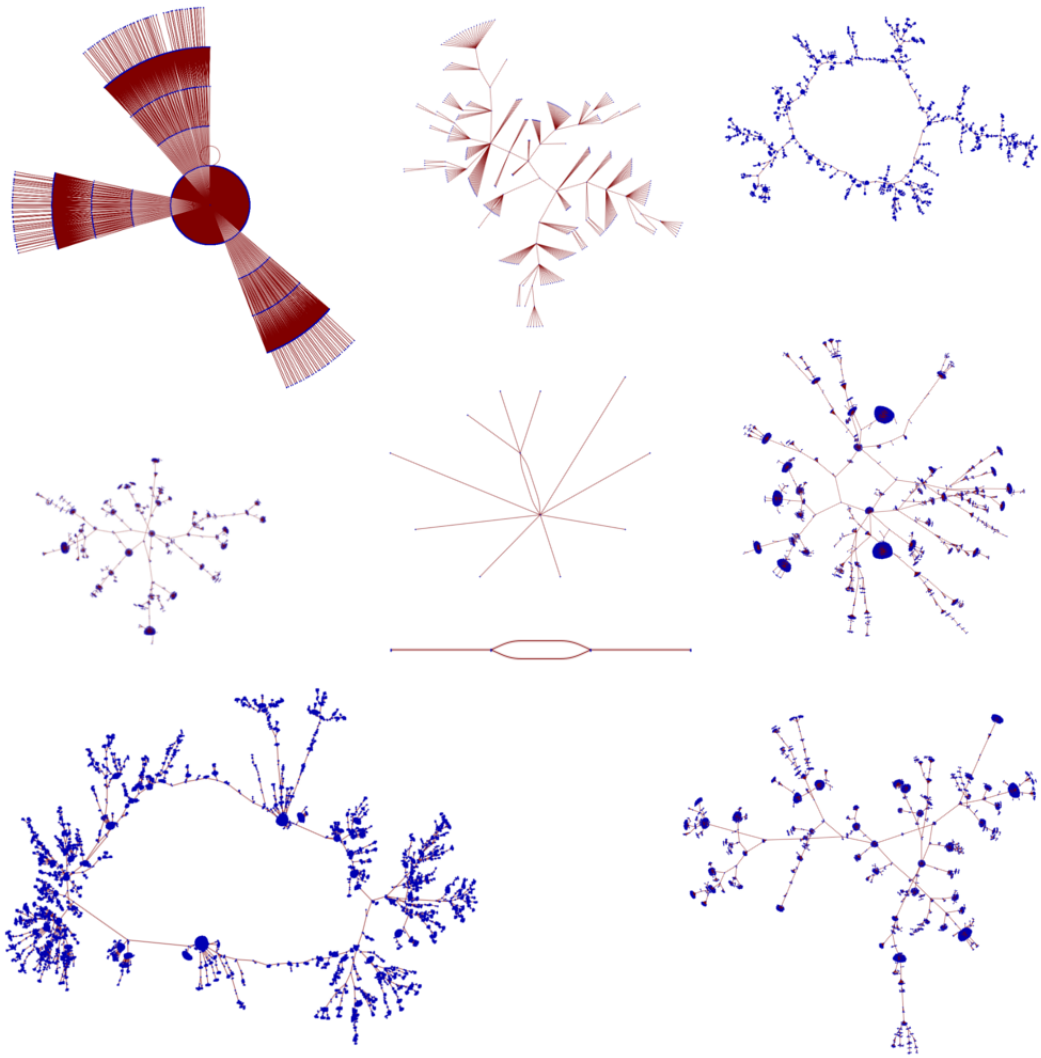


Figura A.6: Attractors with length equal to 21

A.21. Attractors with length equal to 22

As we can see in the following figure, the patterns are no longer so trivial and allow us to better observe their cycles. The number of attractors has increased and in turn the density in the leaves is such that we can only observe a uniform color, but inside there are thousands of initial points towards an attractor state.

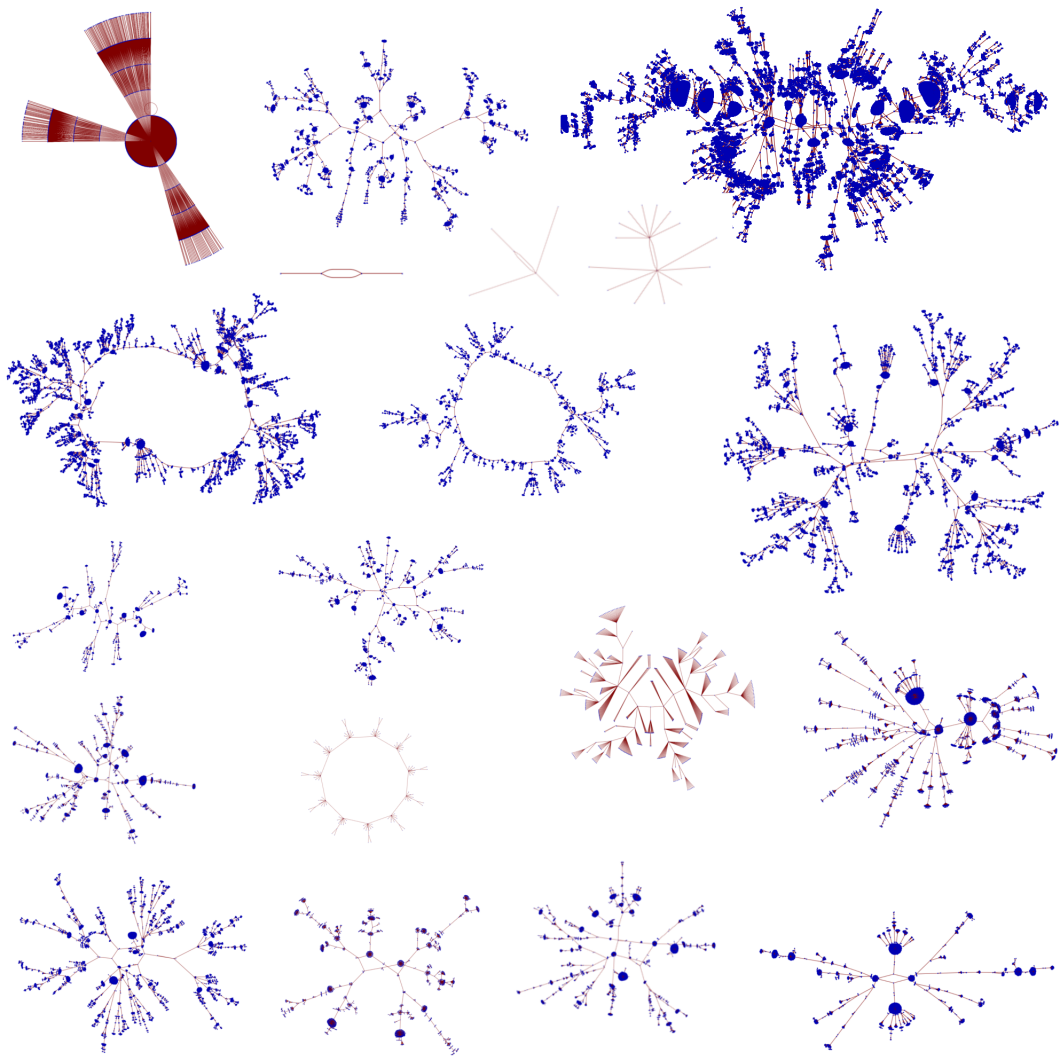


Figura A.7: Attractors with length equal to 22

A.22. Attractors with length equal to 23

Increasingly the number of attractors repeated but with greater Garden of Eden or with greater cycle, but the same roads as the previous ones. This implies that the greater the chain, it contains properties of the previous one:

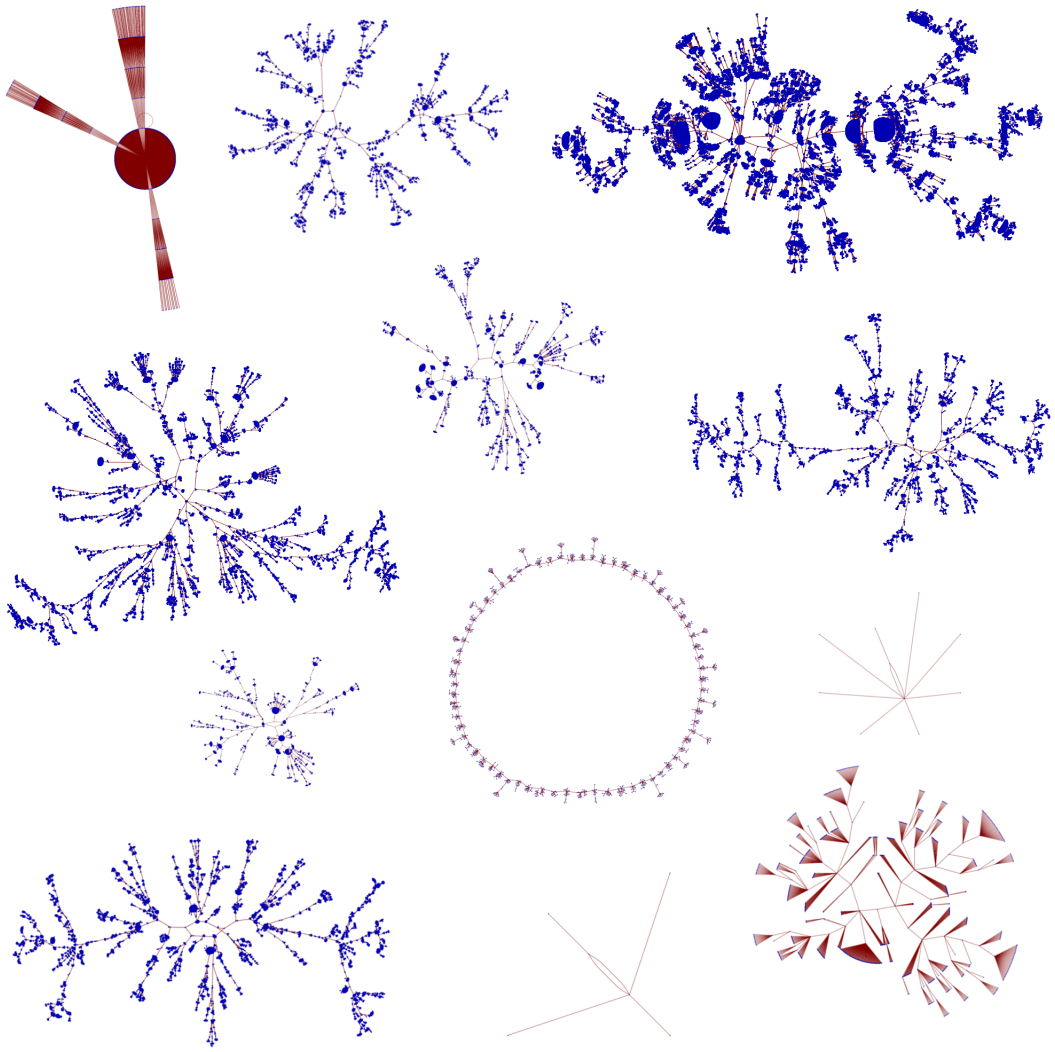


Figura A.8: Attractors with length equal to 23

A.23. Attractors with length equal to 24

The chains of length 24 generate us too many patterns, in total there are 640 attractors within the possible combinations of chains, this is 2^{24} possible states contained in 640 graphs. However, some of these are repeated, so we have taken those that are unique.

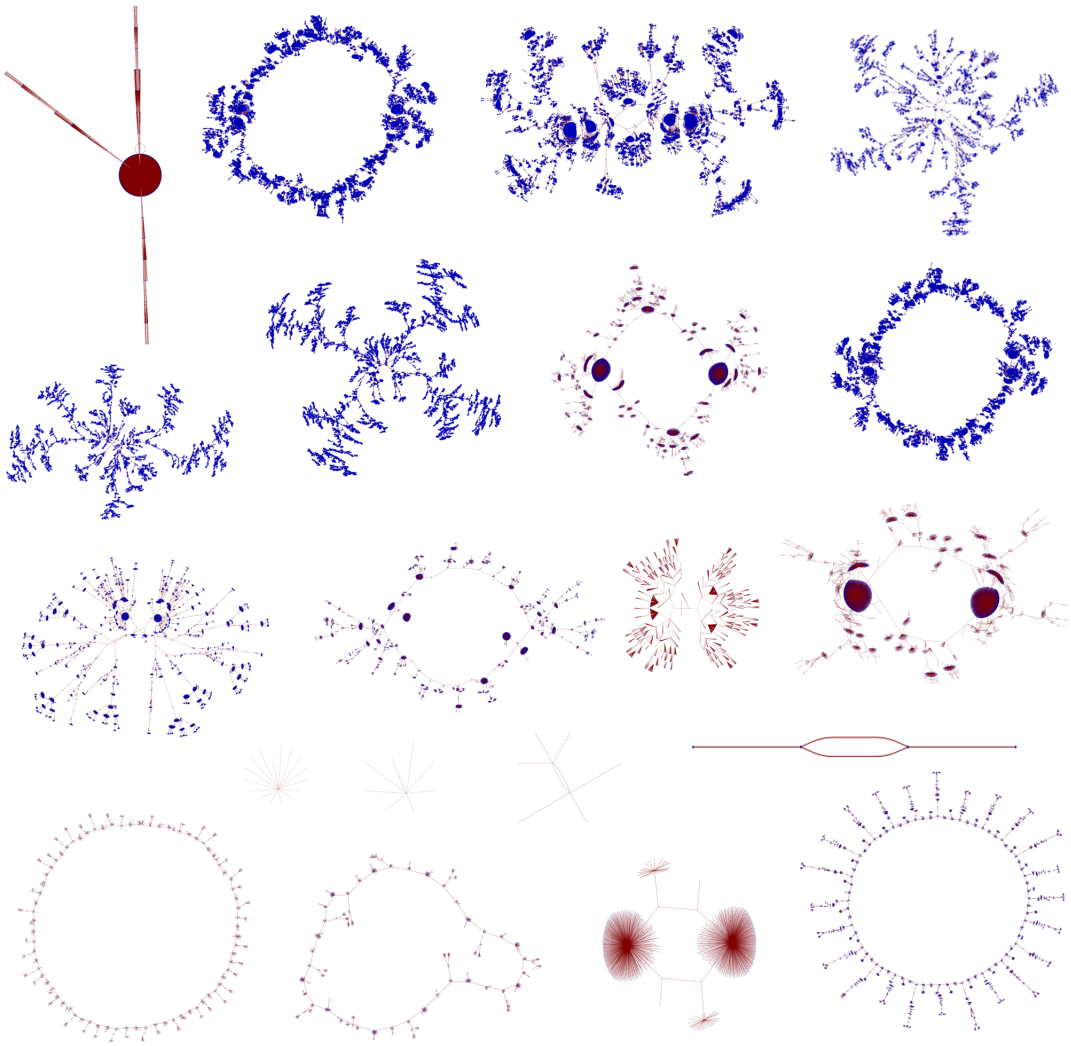


Figura A.9: Attractors with length equal to 24

Apéndice B

Collisions made

This appendix is dedicated to placing all collisions made, regardless of the collision result, in order to show the reader how the various collisions were coded, as well as the complete equation so that it is able to enter it into the simulator and check the results shown in this document.

B.1. Binary collisions

B.1.1. Parameters to define the number of binary collisions to be performed

To define the number of binary collisions to be performed we use set theory to calculate the number of possible combinations among all the elements we have identified. In the special case of binary collisions, the order does not matter how the particles are entered, so we will use a combination without repetitions, and to calculate we use the formula B.1:

$$C = \binom{n}{r} = \frac{n!}{r!(n-r)!} \tag{B.1}$$

Where:

1. n is the number of total elements.

2. r is the amount of elements that we are going to take.

In the case of binary collisions, we have 8 elements identified (6 gliders and 2 still-life, for more details consult the section 4.4), and we will take 2 (binary), so by substituting we obtain that:

$$C = \binom{n}{r} = \frac{8!}{2!(8-2)!} = 28$$

Therefore, we have 28 ways to combine the particles, which would be those shown in the table B.1, however, of these 28 we can only perform 11 for two reasons:

1. **The tile for filtering is different:** Because we have not found a tile for filtering that allows the coexistence of gliders that have a different tile for filtering (3rd tile for filtering), the particles that have different tiles for filtering are impossible to collide. Example of this are the particles g_1 and g_4, because g_1 has the tile for filtering $1(fp_1)$ on the side where it would collide with g_4, which has the 2nd tile for filtering(fp_2).

2. **The particles are parallel:** If the particles move in the same direction, they will never collide because they have the same speed (see section 4.4 for more information). Example of this are the gliders g_6 and g_1, although they have the tile for filtering in common, they move to the right, so they will never collide, another example is the still-life.

So in the table B.1 we will show the combination and if it can be done or not, in case it is not possible one of the two reasons that have been explained will be given.

No.	**Combination**	**Can it be realized?**	**Reason**
1	$g_1 \leftrightarrow g_2$	Yes	
2	$g_1 \leftrightarrow g_3$	No	They are parallel
3	$g_1 \leftrightarrow g_4$	No	Different tile for filtering
4	$g_1 \leftrightarrow g_5$	No	Different tile for filtering and they are parallel
5	$g_1 \leftrightarrow g_6$	Yes	
6	$g_1 \leftrightarrow s_1$	Yes	
7	$g_1 \leftrightarrow s_2$	No	Different tile for filtering

Continued on the next page

Cuadro B.1 – *Continued from previous page*

No.	**Combination**	**Can it be realized?**	**Reason**
8	$g_2 \leftrightarrow g_4$	No	Different tile for filtering and they are parallel
9	$g_2 \leftrightarrow g_6$	No	They are parallel
10	$g_3 \leftrightarrow g_2$	No	Different tile for filtering
11	$g_3 \leftrightarrow g_4$	Yes	
12	$g_3 \leftrightarrow g_5$	No	They are parallel
13	$g_3 \leftrightarrow g_6$	No	Different tile for filtering
14	$g_3 \leftrightarrow s_1$	No	Different tile for filtering
15	$g_3 \leftrightarrow s_2$	Yes	
16	$g_4 \leftrightarrow g_6$	No	They are parallel
17	$g_5 \leftrightarrow g_2$	Yes	
18	$g_5 \leftrightarrow g_4$	No	Different tile for filtering
19	$g_5 \leftrightarrow g_6$	Yes	
20	$g_5 \leftrightarrow s_1$	Yes	
21	$g_5 \leftrightarrow s_2$	No	Different tile for filtering
22	$s_1 \leftrightarrow g_2$	Yes	
23	$s_1 \leftrightarrow g_4$	No	Different tile for filtering
24	$s_1 \leftrightarrow g_6$	Yes	
25	$s_1 \leftrightarrow s_2$	No	They are parallel
26	$s_2 \leftrightarrow g_2$	No	Different tile for filtering
27	$s_2 \leftrightarrow g_4$	Yes	
28	$s_2 \leftrightarrow g_6$	No	Different tile for filtering

Cuadro B.1: Table of possible binary collisions

Thus, we have the total number of combinations that can be simulated, for the case of binaries there are no more possible combinations. Now, for each combination we must determine how many collisions are possible according to the phases that the participating particles possess, for this we will use the formula B.2.

$$Collisions = m * n \tag{B.2}$$

Where

1. m are the number of phases that the first particle that participates has.
2. n are the number of phases that the second particle that participates has.

So, let's take as an example that we will collide g_1 against g_6, both particles have 5 different phases (to see the phases see 4.4), therefore we have:

$$Collisions = 5 * 5 = 25$$

We can collide our system in 25 different ways, regardless of whether the result in any collision is the same or some point of collision is not achievable. In the following sections we will calculate for each of the 11 combinations the number of possible collisions and show the results obtained, as well as the complete equation for its simulation and verification, this equation will be formed by the structure shown in the equation B.3

$$fp_a(f_b) \leftrightarrow g_c(f_d) \leftrightarrow s_e(f)_h = K \tag{B.3}$$

Where:

1. $fp_a(f_b)$ will denote the tile for filtering numbered as a in the phase numbered as b.
2. $g_c(f_d)$ will denote the glider numbered as c in the phase numbered as d.
3. $s_e(f)_h$ will denote the still-life numbered as e in the phase numbered as h.
4. K will denote a combination of gliders and still-life, and it will have the same notation described in the section 4.6.

Finally to divide the collisions, in each section we will place a subsection for each phase of the particle that appears on the left side, in it we will exemplify its interaction with all the phases of the particle on the right side.

B.1.2. Restrictions by dynamics

There are some restrictions due to the dynamics of the tile for filtering at the moment of making the collisions, and that they reduce the amount that are possible to realize, these restrictions are:

1. **The tile for filtering 2 only allows a collision by combination:** This restriction occurs if the elements that participate have the same phases as the tile for filtering. For example, if we try to collide the glider g_3 with the glider g_4, the theoretical amount of possible collisions are 5 * 5 = 25, however, these gliders exist in the tile for filtering 2, which has 5 phases as well, so to maintain the structures and coexist with the tile for filtering, we can only collide with one way and that's when the glider is in phase with the tile for filtering, so we can only collide 5 shapes.

2. **The tile for filtering 2 only allows two collisions per combination:** This case is similar to the first, but only applies to the still-life s_2, since it has 10 phases, the tile for filtering allows us to place it in two different ways. An example is if we collide the glider g_3 with the still-life s_2, theoretically we should give 5 * 10 = 50, however we can only design two collisions that are in phase with the tile for filtering, so in reality we can only simulate 10 shapes.

B.1.3. g1 VS g2

To g_1 against g_2, we have both have 5 phases, so the number of possible collisions we can perform are:

$$Collisions = 5 * 5 = 25 \tag{B.4}$$

g1 in the phase 1

In the figure B.1 we can observe the 5 possible collisions by keeping the glider g_1 phase fixed using phase 1, and rotate the phase of the glider g_2 in the 5 phases that it has.

So the equations that we have as a result are:

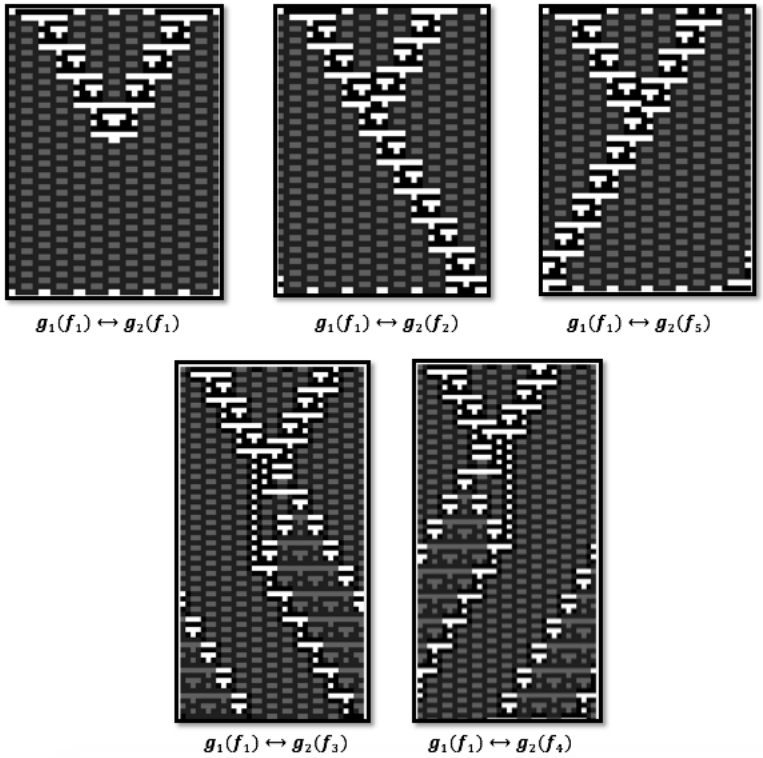


Figura B.1: The 5 different collisions between the glider g_1 in phase 1 and the 5 phases that the glider has g_2

1. $fp_1(f_1) \leftrightarrow g_1(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_1) \leftrightarrow fp_1(f_1) = \emptyset$
2. $fp_1(f_1) \leftrightarrow g_1(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_2) \leftrightarrow fp_1(f_2) = g_1$
3. $fp_1(f_1) \leftrightarrow g_1(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_3) \leftrightarrow fp_1(f_2) = g_3 + g_5$
4. $fp_1(f_1) \leftrightarrow g_1(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_4) \leftrightarrow fp_1(f_1) = g_6 + g_4$
5. $fp_1(f_1) \leftrightarrow g_1(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_5) \leftrightarrow fp_1(f_2) = g_2$

g1 in the phase 2

In the figure B.2 we can observe the 5 possible collisions by keeping the glider g_1 phase fixed using phase 2, and rotate the phase of the glider g_2 in the 5 phases that it has.

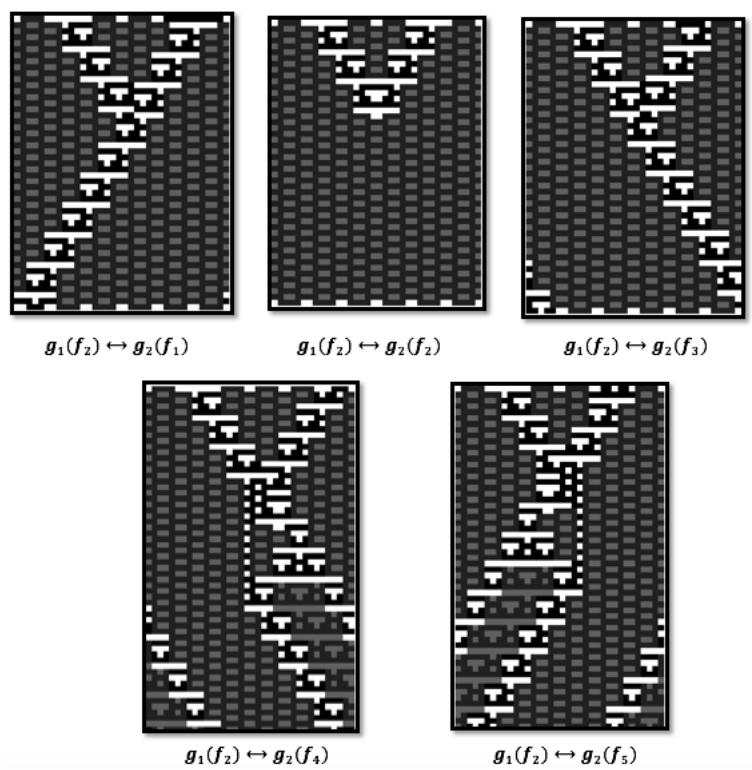


Figura B.2: The 5 different collisions between the glider g_1 in phase 2 and the 5 phases that the glider has g_2

So the equations that we have as a result are:

1. $fp_1(f_2) \leftrightarrow g_1(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_1) \leftrightarrow fp_1(f_1) = g_2$
2. $fp_1(f_2) \leftrightarrow g_1(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_2) \leftrightarrow fp_1(f_2) = \emptyset$
3. $fp_1(f_2) \leftrightarrow g_1(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_3) \leftrightarrow fp_1(f_2) = g_1$
4. $fp_1(f_2) \leftrightarrow g_1(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_4) \leftrightarrow fp_1(f_1) = g_3 + g_5$
5. $fp_1(f_2) \leftrightarrow g_1(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_5) \leftrightarrow fp_1(f_2) = g_6 + g_4$

g1 in the phase 3

In the figure B.3 we can observe the 5 possible collisions by keeping the glider g_1 phase fixed using phase 3, and rotate the phase of the glider g_2 in the 5 phases that it has.

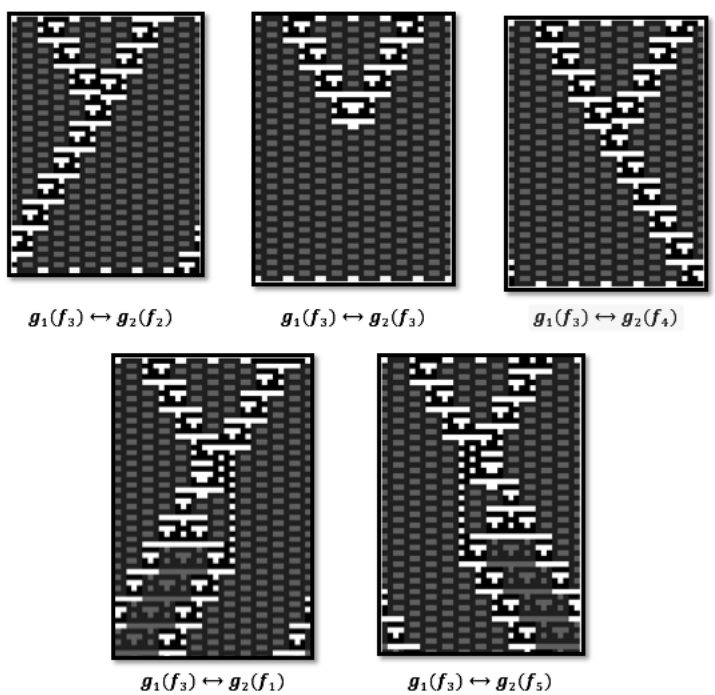


Figura B.3: The 5 different collisions between the glider g_1 in phase 3 and the 5 phases that the glider has g_2

So the equations that we have as a result are:

1. $fp_1(f_2) \leftrightarrow g_1(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_1) \leftrightarrow fp_1(f_1) = g_6 + g_4$
2. $fp_1(f_2) \leftrightarrow g_1(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_2) \leftrightarrow fp_1(f_2) = g_2$
3. $fp_1(f_2) \leftrightarrow g_1(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_3) \leftrightarrow fp_1(f_2) = \emptyset$
4. $fp_1(f_2) \leftrightarrow g_1(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_4) \leftrightarrow fp_1(f_1) = g_1$
5. $fp_1(f_2) \leftrightarrow g_1(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_5) \leftrightarrow fp_1(f_2) = g_3 + g_5$

g1 in the phase 4

In the figure B.4 we can observe the 5 possible collisions by keeping the glider g_1 phase fixed using phase 4, and rotate the phase of the glider g_2 in the 5 phases that it has.

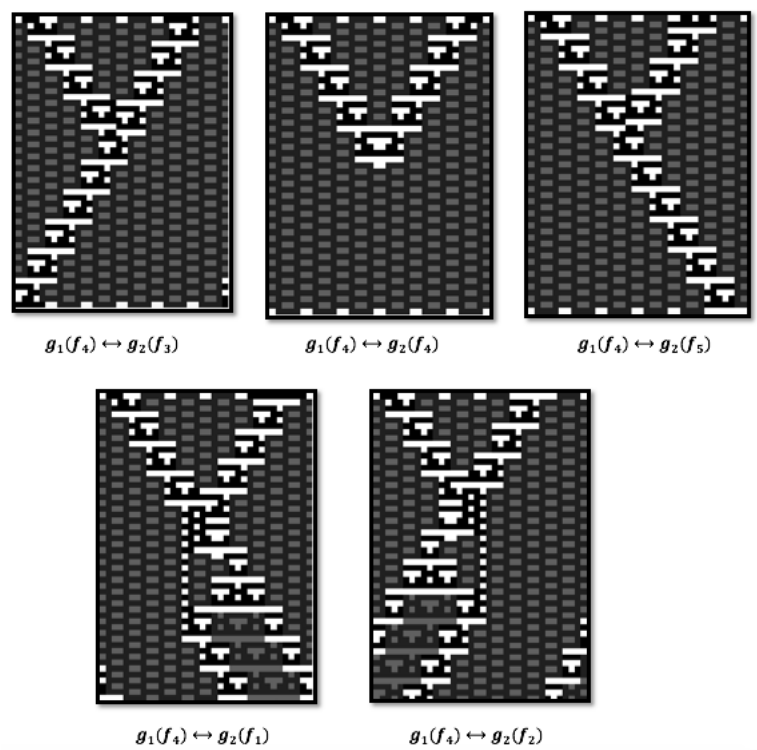


Figura B.4: The 5 different collisions between the glider g_1 in phase 4 and the 5 phases that the glider has g_2

So the equations that we have as a result are:

1. $fp_1(f_1) \leftrightarrow g_1(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_1) \leftrightarrow fp_1(f_1) = g_3 + g_5$
2. $fp_1(f_1) \leftrightarrow g_1(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_2) \leftrightarrow fp_1(f_2) = g_6 + g_4$
3. $fp_1(f_1) \leftrightarrow g_1(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_3) \leftrightarrow fp_1(f_2) = g_2$
4. $fp_1(f_1) \leftrightarrow g_1(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_4) \leftrightarrow fp_1(f_1) = \emptyset$
5. $fp_1(f_1) \leftrightarrow g_1(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_5) \leftrightarrow fp_1(f_2) = g_1$

g1 in the phase 5

In the figure B.5 we can observe the 5 possible collisions by keeping the glider g_1 phase fixed using phase 5, and rotate the phase of the glider g_2 in the 5 phases that it has.

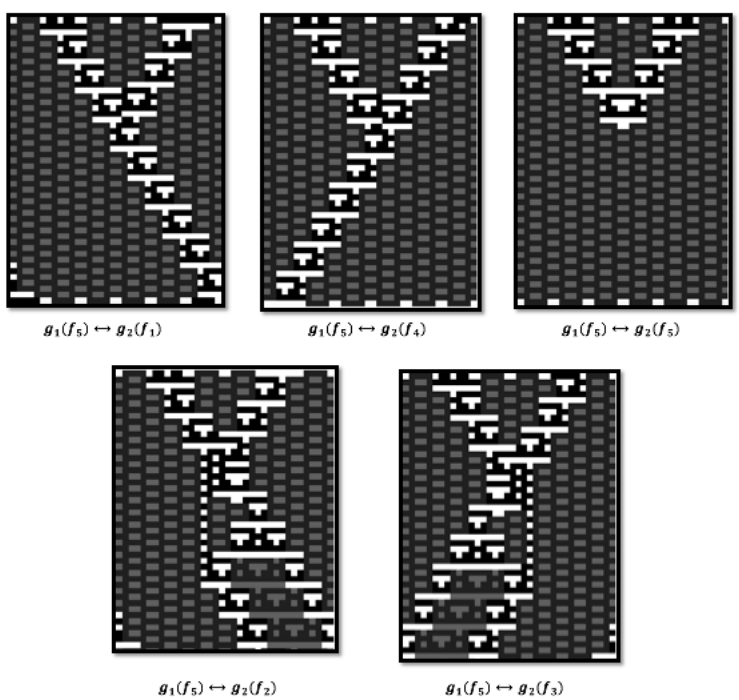


Figura B.5: The 5 different collisions between the glider g_1 in phase 5 and the 5 phases that the glider has g_2

So the equations that we have as a result are:

1. $fp_1(f_2) \leftrightarrow g_1(f_5) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_1) \leftrightarrow fp_1(f_1) = g_1$

2. $fp_1(f_2) \leftrightarrow g_1(f_5) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_2) \leftrightarrow fp_1(f_2) = g_3 + g_5$

3. $fp_1(f_2) \leftrightarrow g_1(f_5) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_3) \leftrightarrow fp_1(f_2) = g_6 + g_4$

4. $fp_1(f_2) \leftrightarrow g_1(f_5) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_4) \leftrightarrow fp_1(f_1) = g_2$

5. $fp_1(f_2) \leftrightarrow g_1(f_5) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_5) \leftrightarrow fp_1(f_2) = \emptyset$

B.1.4. g1 VS g6

To g_1 against g_6, we have both have 5 phases, so the number of possible collisions we can perform are:

$$Collisions = 5 * 5 = 25 \tag{B.5}$$

g1 in the phase 1

In the figure B.6 we can observe the 5 possible collisions by keeping the glider g_1 phase fixed using phase 1, and rotate the phase of the glider g_6 in the 5 phases that it has.

So the equations that we have as a result are:

1. $fp_1(f_1) \leftrightarrow g_1(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_1) \leftrightarrow fp_2(f_1) = g_6$

2. $fp_1(f_1) \leftrightarrow g_1(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_2) \leftrightarrow fp_2(f_2) = g_3^2$

3. $fp_1(f_1) \leftrightarrow g_1(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_3) \leftrightarrow fp_2(f_3) = g_2 + g_6 + g_5 + g_3$

4. $fp_1(f_1) \leftrightarrow g_1(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_4) \leftrightarrow fp_2(f_4) = g_6 + g_5 + g_3$

5. $fp_1(f_1) \leftrightarrow g_1(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_5) \leftrightarrow fp_2(f_5) = g_6$

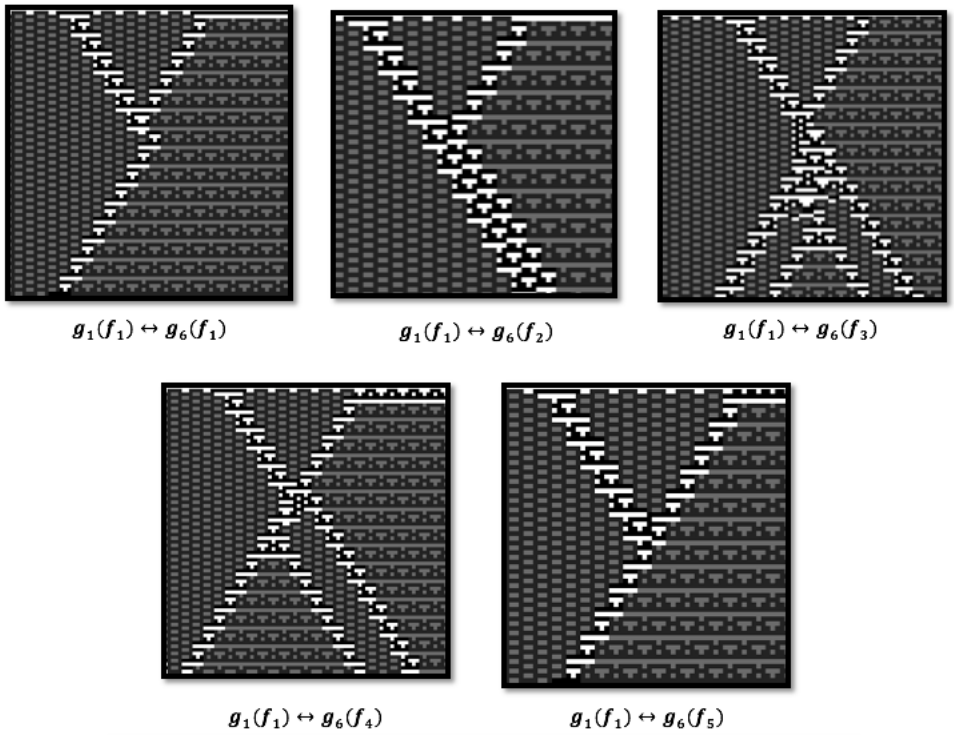


Figura B.6: The 5 different collisions between the glider g_1 in phase 1 and the 5 phases that the glider has g_6

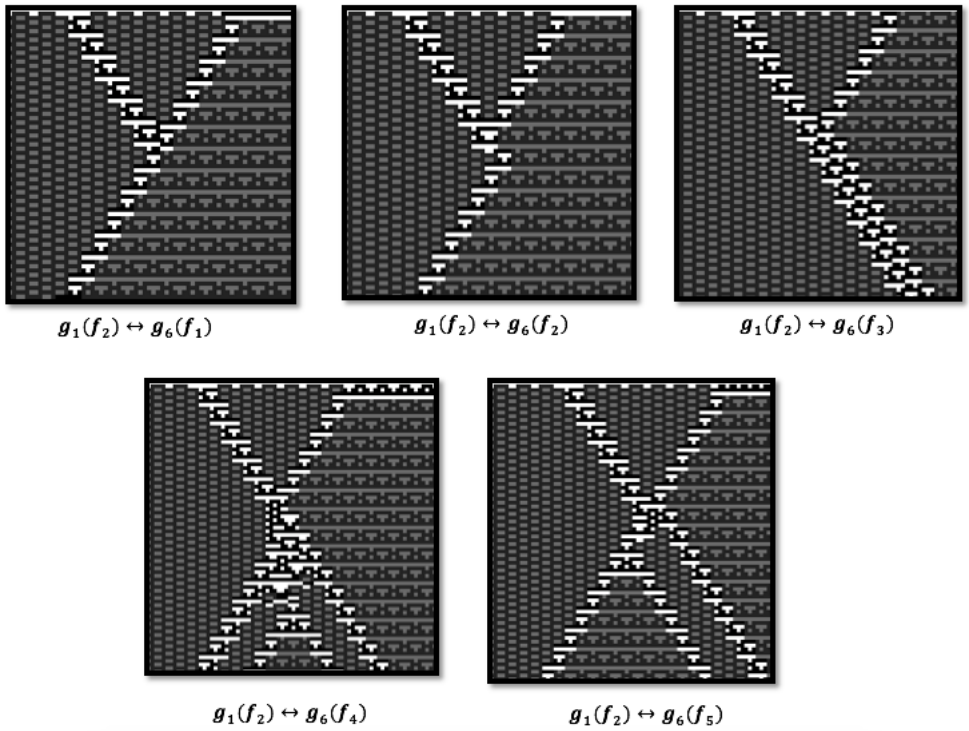


Figura B.7: The 5 different collisions between the glider g_1 in phase 2 and the 5 phases that the glider has g_6

g1 in the phase 2

In the figure B.7 we can observe the 5 possible collisions by keeping the glider g_1 phase fixed using phase 2, and rotate the phase of the glider g_6 in the 5 phases that it has.

So the equations that we have as a result are:

1. $fp_1(f_2) \leftrightarrow g_1(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_1) \leftrightarrow fp_2(f_1) = g_6$
2. $fp_1(f_2) \leftrightarrow g_1(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_2) \leftrightarrow fp_2(f_2) = g_6$
3. $fp_1(f_2) \leftrightarrow g_1(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_3) \leftrightarrow fp_2(f_3) = g_3^2$
4. $fp_1(f_2) \leftrightarrow g_1(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_4) \leftrightarrow fp_2(f_4) = g_2 + g_6 + g_5 + g_3$
5. $fp_1(f_2) \leftrightarrow g_1(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_5) \leftrightarrow fp_2(f_5) = g_6 + g_5 + g_3$

g1 in the phase 3

In the figure B.8 we can observe the 5 possible collisions by keeping the glider g_1 phase fixed using phase 3, and rotate the phase of the glider g_6 in the 5 phases that it has.

So the equations that we have as a result are:

1. $fp_1(f_2) \leftrightarrow g_1(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_1) \leftrightarrow fp_2(f_1) = g_6 + g_5 + g_3$
2. $fp_1(f_2) \leftrightarrow g_1(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_2) \leftrightarrow fp_2(f_2) = g_6$
3. $fp_1(f_2) \leftrightarrow g_1(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_3) \leftrightarrow fp_2(f_3) = g_6$
4. $fp_1(f_2) \leftrightarrow g_1(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_4) \leftrightarrow fp_2(f_4) = g_3^2$
5. $fp_1(f_2) \leftrightarrow g_1(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_5) \leftrightarrow fp_2(f_5) = g_2 + g_6 + g_5 + g_3$

g1 in the phase 4

In the figure B.9 we can observe the 5 possible collisions by keeping the glider g_1 phase fixed using phase 4, and rotate the phase of the glider g_6 in the 5 phases that it has.

So the equations that we have as a result are:

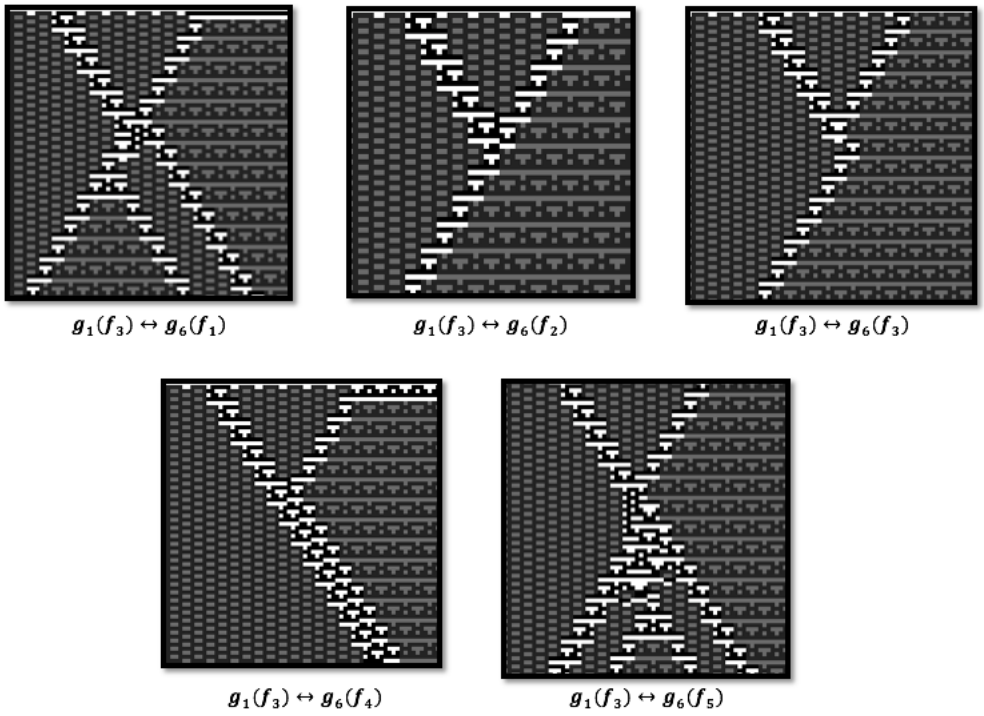


Figura B.8: The 5 different collisions between the glider g_1 in phase 3 and the 5 phases that the glider has g_6

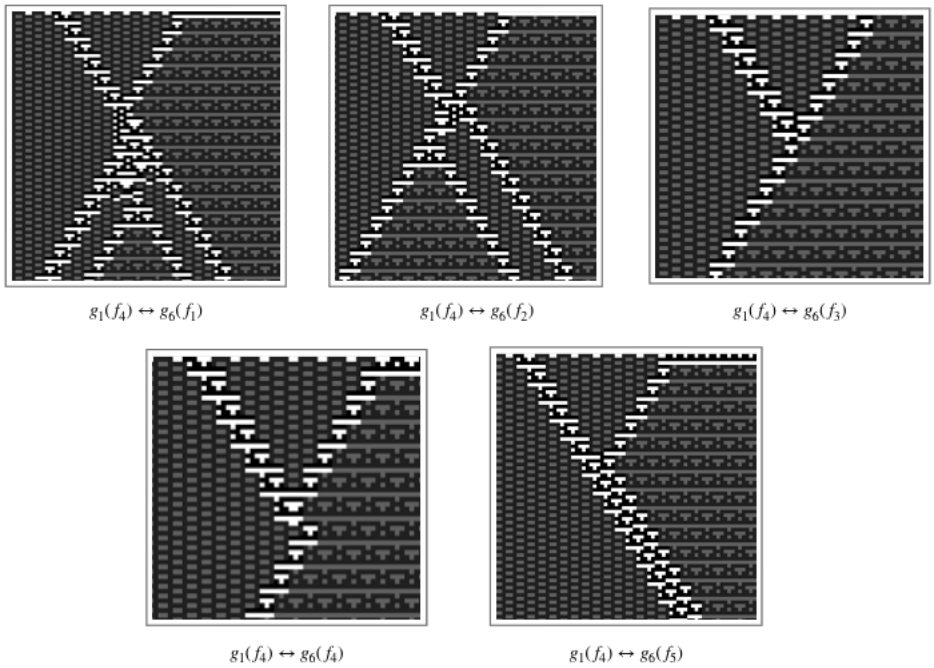


Figura B.9: The 5 different collisions between the glider g_1 in phase 4 and the 5 phases that the glider has g_6

1. $fp_1(f_1) \leftrightarrow g_1(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_1) \leftrightarrow fp_2(f_1) = g_2 + g_6 + g_5 + g_3$
2. $fp_1(f_1) \leftrightarrow g_1(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_2) \leftrightarrow fp_2(f_2) = g_6 + g_5 + g_3$
3. $fp_1(f_1) \leftrightarrow g_1(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_3) \leftrightarrow fp_2(f_3) = g_6$
4. $fp_1(f_1) \leftrightarrow g_1(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_4) \leftrightarrow fp_2(f_4) = g_6$
5. $fp_1(f_1) \leftrightarrow g_1(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_5) \leftrightarrow fp_2(f_5) = g_3^2$

g1 in the phase 5

In the figure B.10 we can observe the 5 possible collisions by keeping the glider g_1 phase fixed using phase 5, and rotate the phase of the glider g_6 in the 5 phases that it has.

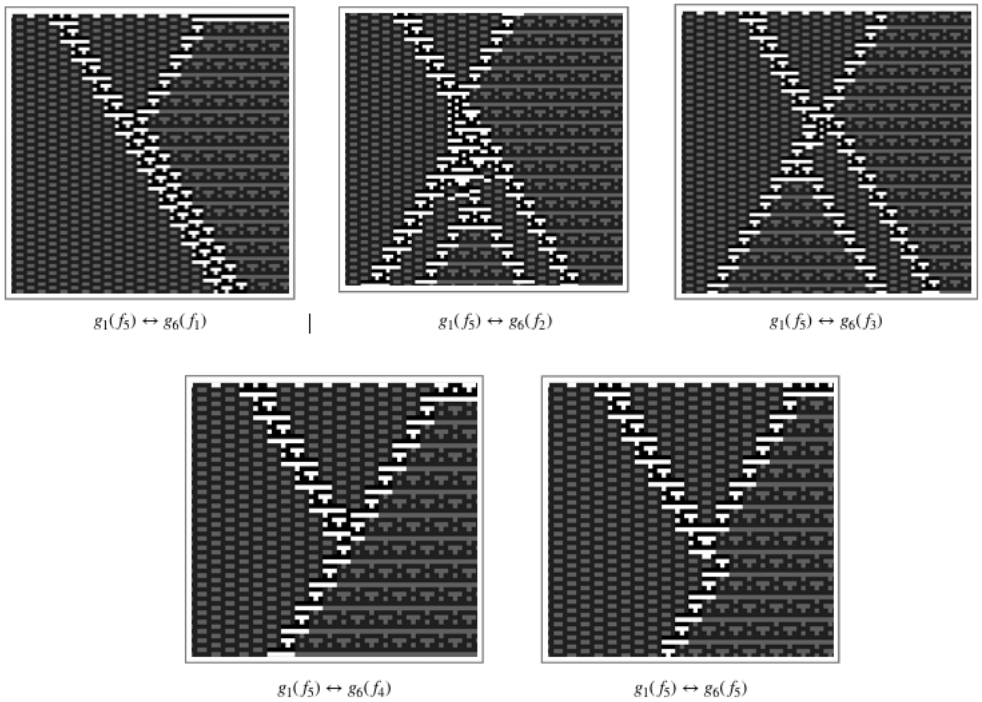


Figura B.10: The 5 different collisions between the glider g_1 in phase 5 and the 5 phases that the glider has g_6

So the equations that we have as a result are:

1. $fp_1(f_2) \leftrightarrow g_1(f_5) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_1) \leftrightarrow fp_2(f_1) = g_3^2$
2. $fp_1(f_2) \leftrightarrow g_1(f_5) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_2) \leftrightarrow fp_2(f_2) = g_2 + g_6 + g_5 + g_3$
3. $fp_1(f_2) \leftrightarrow g_1(f_5) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_3) \leftrightarrow fp_2(f_3) = g_6 + g_5 + g_3$
4. $fp_1(f_2) \leftrightarrow g_1(f_5) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_4) \leftrightarrow fp_2(f_4) = g_6$
5. $fp_1(f_2) \leftrightarrow g_1(f_5) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_5) \leftrightarrow fp_2(f_5) = g_6$

B.1.5. g1 VS s1

To g_1 against s_1, we have that g_1 has 5 phases, while s_1 only has 2, so the number of possible collisions are:

$$Collisions = 5 * 2 = 10 \tag{B.6}$$

g1 in the phase 1

In the figure B.11 we can observe the 5 possible collisions by keeping the glider g_1 phase fixed using phase 1, and rotate the phase of the still-life s_1 in the 2 phases that it has.

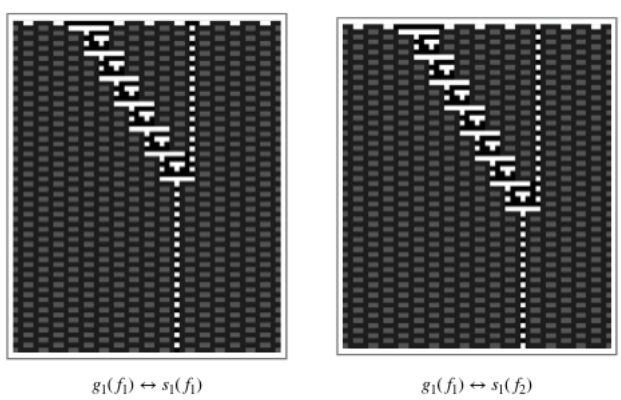


Figura B.11: The 5 different collisions between the glider g_1 in phase 1 and the 2 phases that the still-life s_1 has

So the equations that we have as a result are:

1. $fp_1(f_1) \leftrightarrow g_1(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_1) = s_1$
2. $fp_1(f_1) \leftrightarrow g_1(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_2) = s_1$

g1 in the phase 2

in the figure B.12 we can observe the 5 possible collisions by keeping the glider g_1 phase fixed using phase 2, and rotate the phase of the still-life s_1 in the 2 phases that it has.

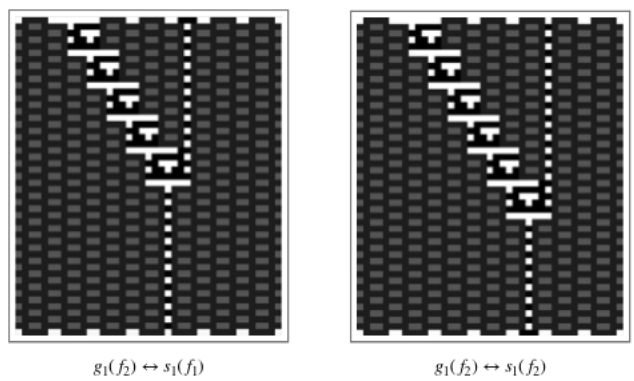


Figura B.12: The 5 different collisions between the glider g_1 in phase 2 and the 2 phases that the still-life s_1 has

So the equations that we have as a result are:

1. $fp_1(f_2) \leftrightarrow g_1(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_1) = s_1$
2. $fp_1(f_2) \leftrightarrow g_1(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_2) = s_1$

g1 in the phase 3

In the figure B.13 we can observe the 5 possible collisions by keeping the glider g_1 phase fixed using phase 3, and rotate the phase of the still-life s_1 in the 2 phases that it has.

So the equations that we have as a result are:

1. $fp_1(f_2) \leftrightarrow g_1(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_1) = s_1$
2. $fp_1(f_2) \leftrightarrow g_1(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_2) = s_1$

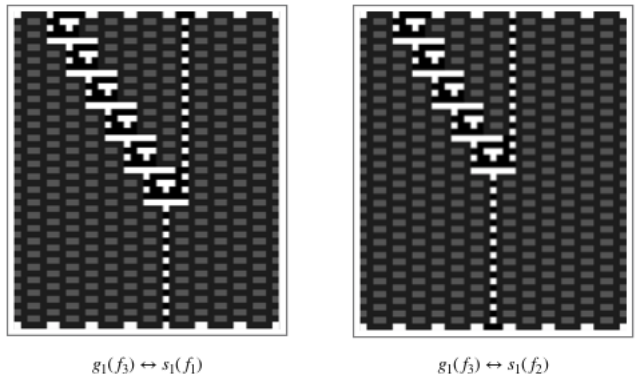


Figura B.13: The 5 different collisions between the glider g_1 in phase 3 and the 2 phases that the still-life s_1 has

g1 in the phase 4

In the figure B.14 we can observe the 5 possible collisions by keeping the glider g_1 phase fixed using phase 4, and rotate the phase of the still-life s_1 in the 2 phases that it has.

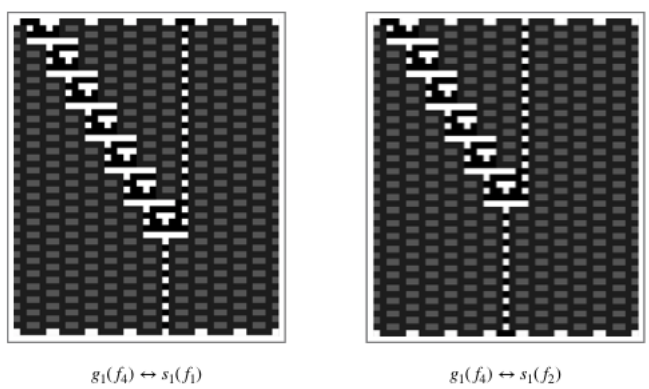


Figura B.14: The 5 different collisions between the glider g_1 in phase 4 and the 2 phases that the still-life s_1 has

So the equations that we have as a result are:

1. $fp_1(f_1) \leftrightarrow g_1(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_1) = s_1$

2. $fp_1(f_1) \leftrightarrow g_1(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_2) = s_1$

g1 in the phase 5

In the figure B.15 we can observe the 5 possible collisions by keeping the glider g_1 phase fixed using phase 5, and rotate the phase of the still-life s_1 in the 2 phases that it has.

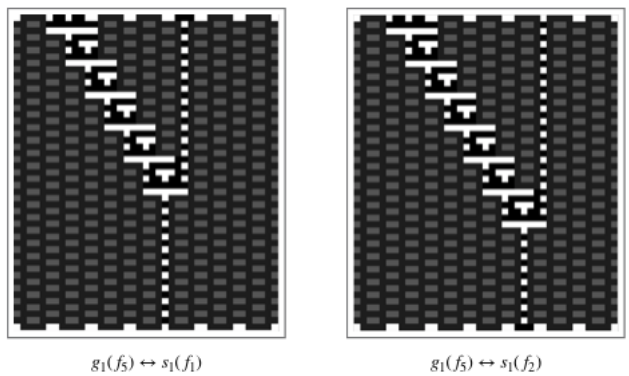


Figura B.15: The 5 different collisions between the glider g_1 in phase 5 and the 2 phases that the still-life s_1 has

So the equations that we have as a result are:

1. $fp_1(f_2) \leftrightarrow g_1(f_5) \leftrightarrow fp_1(f_1) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_1) = s_1$
2. $fp_1(f_2) \leftrightarrow g_1(f_5) \leftrightarrow fp_1(f_1) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_2) = s_1$

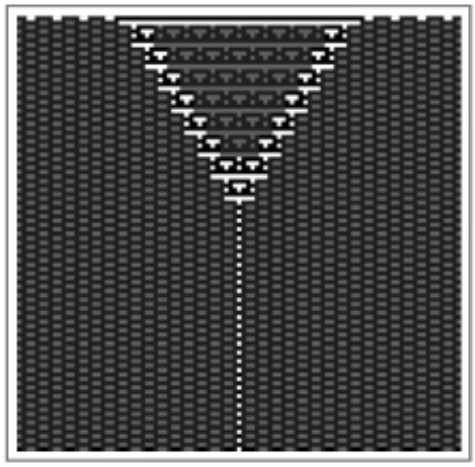
B.1.6. g3 VS g4

To g_3 against g_4, we both have 5 phases, however it is necessary to apply the restriction criterion number 1 indicated in B.1.2, so the possible collisions are:

$$\begin{aligned} Theoretical\ collisions &= 5 * 5 = 25 \\ Real\ collisions &= 5 * 1 = 5 \end{aligned} \tag{B.7}$$

g3 in the phase 1

In the figure B.16 we can observe the only possible collision by keeping the glider g_3 phase fixed using phase 1, and use phase 1 of the glider g_4:



$g_3(f_1) \leftrightarrow g_4(f_1)$

Figura B.16: The possible collision between the glider g_3 in phase 1 and the glider g_4 in phase 1

So, the equation we have as a result is:

1. $fp_1(f_1) \leftrightarrow g_3(f_1) \leftrightarrow fp_2(f_1) \leftrightarrow g_4(f_1) \leftrightarrow fp_1(f_1) = s_1$

g3 in the phase 2

En la figura B.17 we can observe the only possible collision by keeping the glider g_3 phase fixed using phase 2, and use phase 2 of the glider g_4:

So, the equation we have as a result is:

1. $fp_1(f_2) \leftrightarrow g_3(f_2) \leftrightarrow fp_2(f_2) \leftrightarrow g_4(f_2) \leftrightarrow fp_1(f_2) = s_1$

g3 in the phase 3

In the figure B.18 we can observe the only possible collision by keeping the glider g_3 phase fixed using phase 3, and use phase 3 of the glider g_4:

So, the equation we have as a result is:

1. $fp_1(f_2) \leftrightarrow g_3(f_3) \leftrightarrow fp_2(f_3) \leftrightarrow g_4(f_3) \leftrightarrow fp_1(f_2) = s_1$

$g_3(f_2) \leftrightarrow g_4(f_2)$

Figura B.17: The possible collision between the glider g_3 in phase 2 and the glider g_4 in phase 2

$g_3(f_3) \leftrightarrow g_4(f_3)$

Figura B.18: The possible collision between the glider g_3 in phase 3 and the glider g_4 in phase 3

g3 in the phase 4

In the figure B.19 we can observe the only possible collision by keeping the glider g_3 phase fixed using phase 4, and use phase 4 of the glider g_4:

$g_3(f_4) \leftrightarrow g_4(f_4)$

Figura B.19: The possible collision between the glider g_3 in phase 4 and the glider g_4 in phase 4

So, the equation we have as a result is:

1. $fp_1(f_1) \leftrightarrow g_3(f_4) \leftrightarrow fp_2(f_4) \leftrightarrow g_4(f_4) \leftrightarrow fp_1(f_1) = s_1$

g3 in the phase 5

In the figure B.20 we can observe the only possible collision by keeping the glider g_3 phase fixed using phase 5, and use phase 5 of the glider g_4:

So, the equation we have as a result is:

1. $fp_1(f_2) \leftrightarrow g_3(f_5) \leftrightarrow fp_2(f_5) \leftrightarrow g_4(f_5) \leftrightarrow fp_1(f_2) = s_1$

B.1.7. g3 VS s2

To g_3 against s_2, we have that g_3 has 5 phases and s_2 has 10, however criteria 1 and 2 of the section B.1.2 apply, so the number of collisions that can be made are:

$g_3(f_5) \leftrightarrow g_4(f_5)$

Figura B.20: The possible collision between the glider g_3 in phase 5 and the glider g_4 in phase 5

$$\begin{aligned} Theoretical\ collisions &= 5 * 10 = 50 \\ Real\ collisions &= 5 * 2 = 10 \end{aligned} \tag{B.8}$$

g3 in the phase 1

In the figure B.21 we can observe the two possible collisions by keeping the glider g_3 phase fixed using phase 1, and use phase 1 and 6 of the still-life s_2:

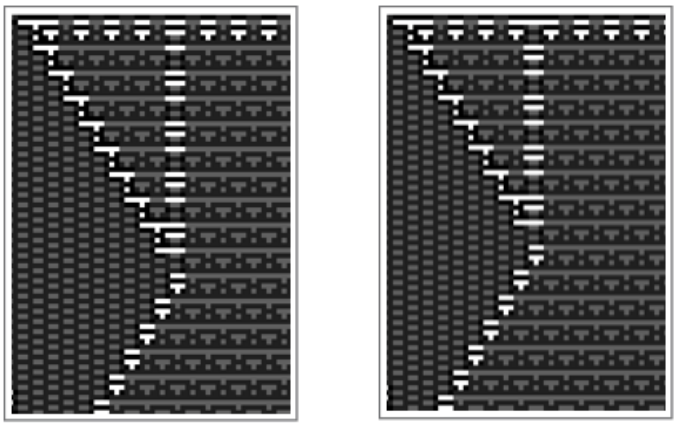
So, the equations we have as a result are:

1. $fp_1(f_1) \leftrightarrow g_3(f_1) \leftrightarrow fp_2(f_1) \leftrightarrow s_2(f_1) \leftrightarrow fp_2(f_1) = g_6$
2. $fp_1(f_1) \leftrightarrow g_3(f_1) \leftrightarrow fp_2(f_1) \leftrightarrow s_2(f_6) \leftrightarrow fp_2(f_1) = g_6$

g3 in the phase 2

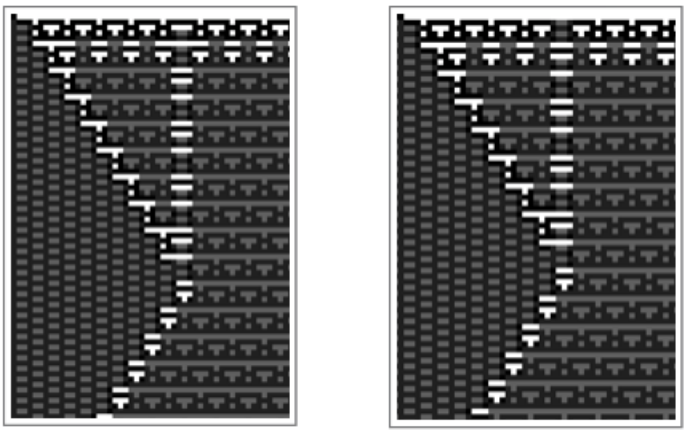
In the figure B.22 we can observe the two possible collisions by keeping the glider g_3 phase fixed using phase 2, and use phase 2 and 7 of the still-life s_2:

So, the equations we have as a result are:



$g_3(f_1) \leftrightarrow s_2(f_1)$ $g_3(f_1) \leftrightarrow s_2(f_6)$

Figura B.21: The possible collisions between the glider g_3 in phase 1 and the still-life s_2 in phases 1 and 6



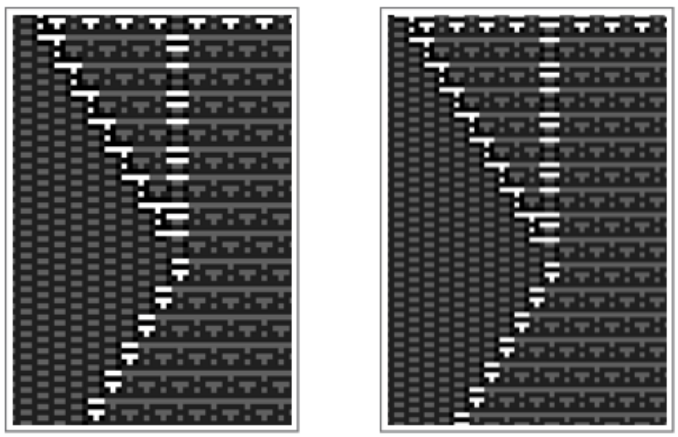
$g_3(f_2) \leftrightarrow s_2(f_2)$ $g_3(f_2) \leftrightarrow s_2(f_7)$

Figura B.22: The possible collisions between the glider g_3 in phase 2 and the still-life s_2 in phases 2 and 7

1. $fp_1(f_2) \leftrightarrow g_3(f_2) \leftrightarrow fp_2(f_2) \leftrightarrow s_2(f_2) \leftrightarrow fp_2(f_2) = g_6$
2. $fp_1(f_2) \leftrightarrow g_3(f_2) \leftrightarrow fp_2(f_2) \leftrightarrow s_2(f_7) \leftrightarrow fp_2(f_2) = g_6$

g3 in the phase 3

In the figure B.23 we can observe the two possible collisions by keeping the glider g_3 phase fixed using phase 3, and use phase 3 and 8 of the still-life s_2:



$g_3(f_3) \leftrightarrow s_2(f_3)$ $g_3(f_3) \leftrightarrow s_2(f_8)$

Figura B.23: The possible collisions between the glider g_3 in phase 3 and the still-life s_2 in phases 3 and 8

So, the equations we have as a result are:

1. $fp_1(f_2) \leftrightarrow g_3(f_3) \leftrightarrow fp_2(f_3) \leftrightarrow s_2(f_3) \leftrightarrow fp_2(f_3) = g_6$
2. $fp_1(f_2) \leftrightarrow g_3(f_3) \leftrightarrow fp_2(f_3) \leftrightarrow s_2(f_8) \leftrightarrow fp_2(f_3) = g_6$

g3 in the phase 4

In the figure B.24 we can observe the two possible collisions by keeping the glider g_3 phase fixed using phase 4, and use phase 4 and 9 of the still-life s_2:

So, the equations we have as a result are:

1. $fp_1(f_1) \leftrightarrow g_3(f_4) \leftrightarrow fp_2(f_4) \leftrightarrow s_2(f_4) \leftrightarrow fp_2(f_4) = g_6$
2. $fp_1(f_1) \leftrightarrow g_3(f_4) \leftrightarrow fp_2(f_4) \leftrightarrow s_2(f_9) \leftrightarrow fp_2(f_4) = g_6$

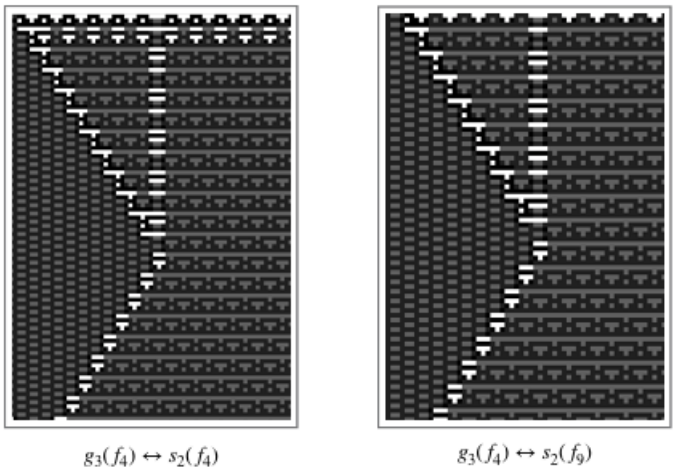


Figura B.24: The possible collisions between the glider g_3 in phase 4 and the still-life s_2 in phases 4 and 9

g3 in the phase 5

In the figure B.25 we can observe the two possible collisions by keeping the glider g_3 phase fixed using phase 5, and use phase 5 and 10 of the still-life s_2:

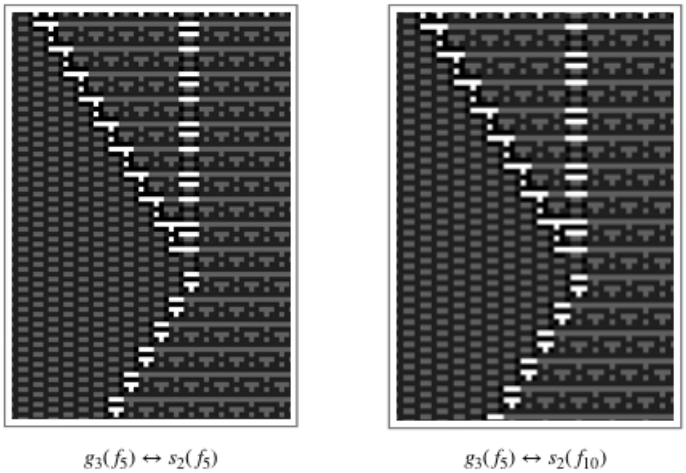


Figura B.25: The possible collisions between the glider g_3 in phase 5 and the still-life s_2 in phases 5 and 10

So, the equations we have as a result are:

1. $fp_1(f_2) \leftrightarrow g_3(f_5) \leftrightarrow fp_2(f_5) \leftrightarrow s_2(f_5) \leftrightarrow fp_2(f_5) = g_6$
2. $fp_1(f_2) \leftrightarrow g_3(f_5) \leftrightarrow fp_2(f_5) \leftrightarrow s_2(f_{10}) \leftrightarrow fp_2(f_5) = g_6$

B.1.8. g5 VS g2

To g_5 against g_2, we have both have 5 phases, so the number of collisions that we can perform are:

$$Collisions = 5 * 5 = 25 \tag{B.9}$$

g5 in the phase 1

In the figure B.26 we can observe the 5 possible collisions by keeping the glider g_5 phase fixed using phase 1, and rotate the phase of the glider g_2 in the 5 phases that it has.

So the equations that we have as a result are:

1. $fp_2(f_1) \leftrightarrow g_5(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_1) \leftrightarrow fp_1(f_1) = g_5$
2. $fp_2(f_1) \leftrightarrow g_5(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_2) \leftrightarrow fp_1(f_2) = g_5$
3. $fp_2(f_1) \leftrightarrow g_5(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_3) \leftrightarrow fp_1(f_2) = g_4 + g_6 + g_5$
4. $fp_2(f_1) \leftrightarrow g_5(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_4) \leftrightarrow fp_1(f_1) = g_4 + g_6 + g_5 + g_2$
5. $fp_2(f_1) \leftrightarrow g_5(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_5) \leftrightarrow fp_1(f_2) = g_4^2$

g5 in the phase 2

In the figure B.27 we can observe the 5 possible collisions by keeping the glider g_5 phase fixed using phase 2, and rotate the phase of the glider g_2 in the 5 phases that it has.

So the equations that we have as a result are:

1. $fp_2(f_2) \leftrightarrow g_5(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_1) \leftrightarrow fp_1(f_1) = g_4^2$

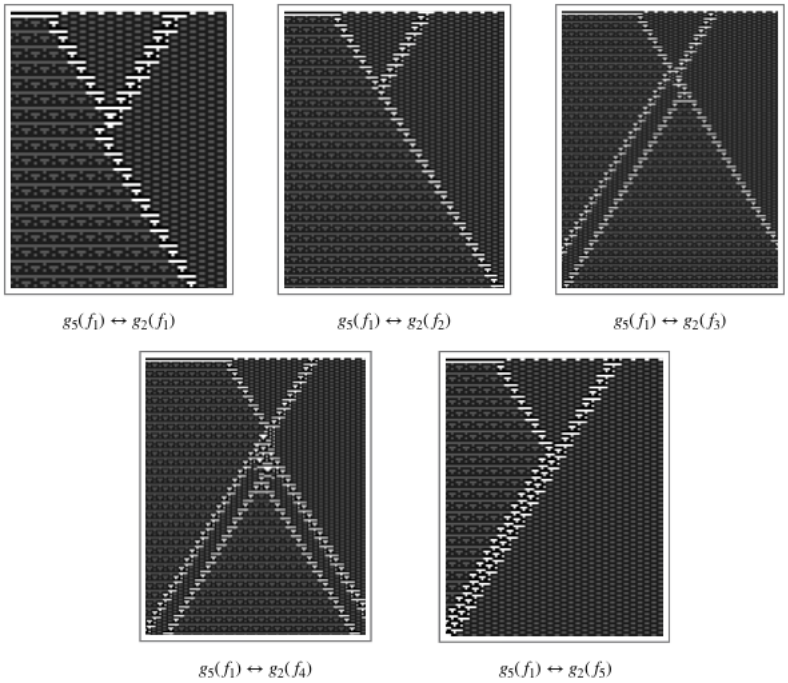


Figura B.26: The 5 different collisions between the glider g_5 in phase 1 and the 5 phases that the glider g_2 has

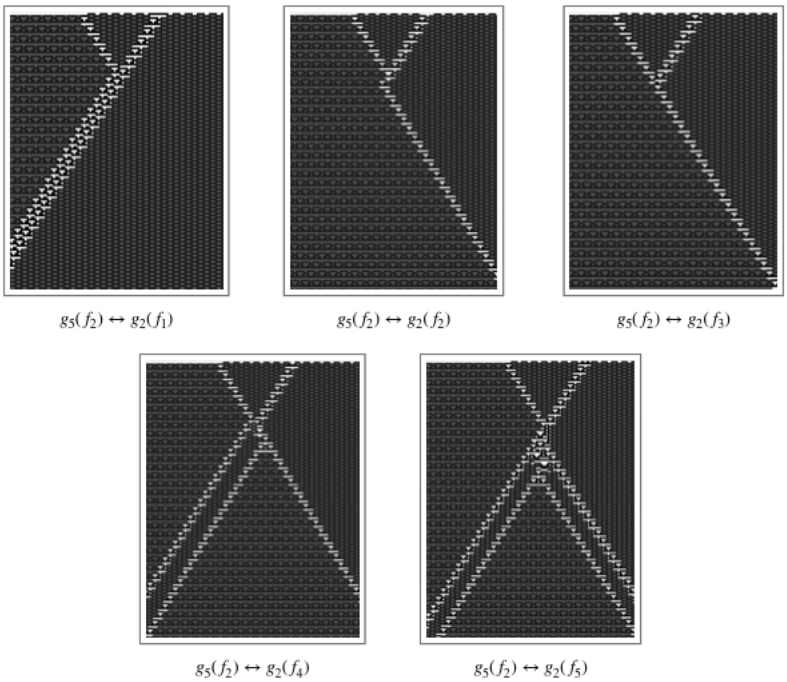


Figura B.27: The 5 different collisions between the glider g_5 in phase 2 and the 5 phases that the glider g_2 has

2. $fp_2(f_2) \leftrightarrow g_5(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_2) \leftrightarrow fp_1(f_2) = g_5$
3. $fp_2(f_2) \leftrightarrow g_5(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_3) \leftrightarrow fp_1(f_2) = g_5$
4. $fp_2(f_2) \leftrightarrow g_5(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_4) \leftrightarrow fp_1(f_1) = g_4 + g_6 + g_5$
5. $fp_2(f_2) \leftrightarrow g_5(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_5) \leftrightarrow fp_1(f_2) = g_4 + g_6 + g_5 + g_2$

g5 in the phase 3

In the figure B.28 we can observe the 5 possible collisions by keeping the glider g_5 phase fixed using phase 3, and rotate the phase of the glider g_2 in the 5 phases that it has.

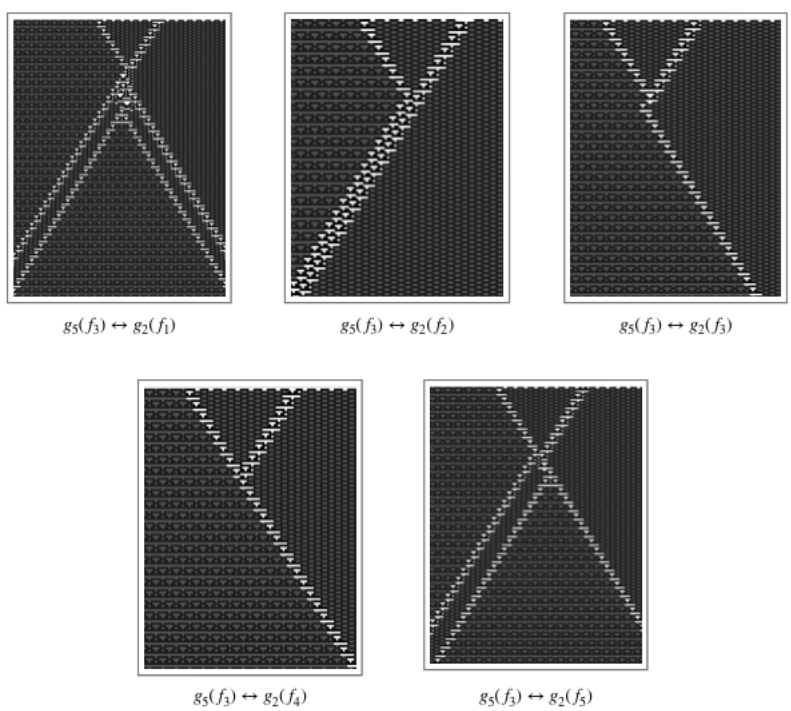


Figura B.28: The 5 different collisions between the glider g_5 in phase 3 and the 5 phases that the glider g_2 has

So the equations that we have as a result are:

1. $fp_2(f_3) \leftrightarrow g_5(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_1) \leftrightarrow fp_1(f_1) = g_4 + g_6 + g_5 + g_2$

2. $fp_2(f_3) \leftrightarrow g_5(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_2) \leftrightarrow fp_1(f_2) = g_4^2$
3. $fp_2(f_3) \leftrightarrow g_5(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_3) \leftrightarrow fp_1(f_2) = g_5$
4. $fp_2(f_3) \leftrightarrow g_5(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_4) \leftrightarrow fp_1(f_1) = g_5$
5. $fp_2(f_3) \leftrightarrow g_5(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_5) \leftrightarrow fp_1(f_2) = g_4 + g_6 + g_5$

g5 in the phase 4

In the figure B.29 we can observe the 5 possible collisions by keeping the glider g_5 phase fixed using phase 4, and rotate the phase of the glider g_2 in the 5 phases that it has.

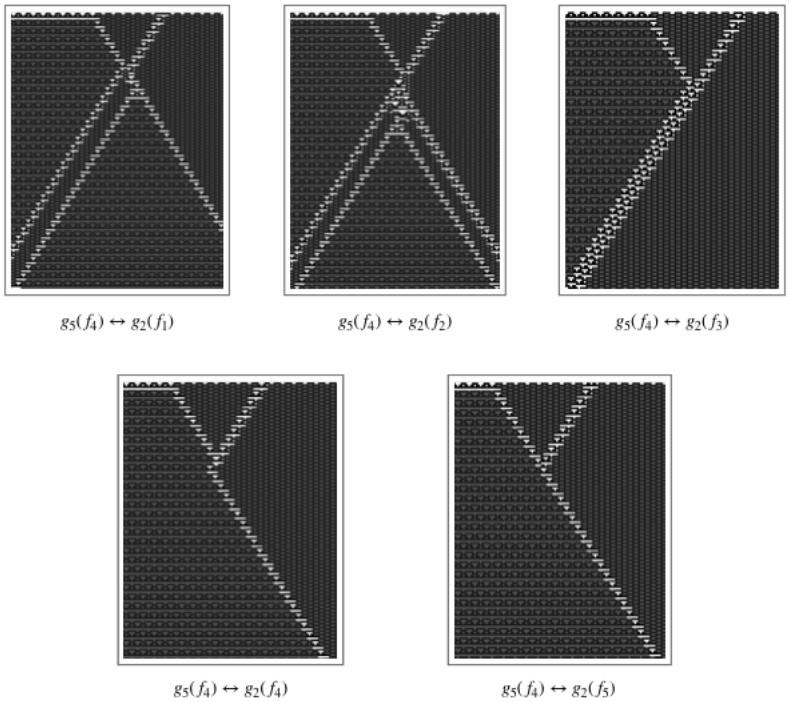


Figura B.29: The 5 different collisions between the glider g_5 in phase 4 and the 5 phases that the glider g_2 has

So the equations that we have as a result are:

1. $fp_2(f_4) \leftrightarrow g_5(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_1) \leftrightarrow fp_1(f_1) = g_4 + g_6 + g_5$

2. $fp_2(f_4) \leftrightarrow g_5(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_2) \leftrightarrow fp_1(f_2) = g_4 + g_6 + g_5 + g_2$
3. $fp_2(f_4) \leftrightarrow g_5(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_3) \leftrightarrow fp_1(f_2) = g_4^2$
4. $fp_2(f_4) \leftrightarrow g_5(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_4) \leftrightarrow fp_1(f_1) = g_5$
5. $fp_2(f_4) \leftrightarrow g_5(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_5) \leftrightarrow fp_1(f_2) = g_5$

g5 in the phase 5

In the figure B.30 we can observe the 5 possible collisions by keeping the glider g_5 phase fixed using phase 5, and rotate the phase of the glider g_2 in the 5 phases that it has.

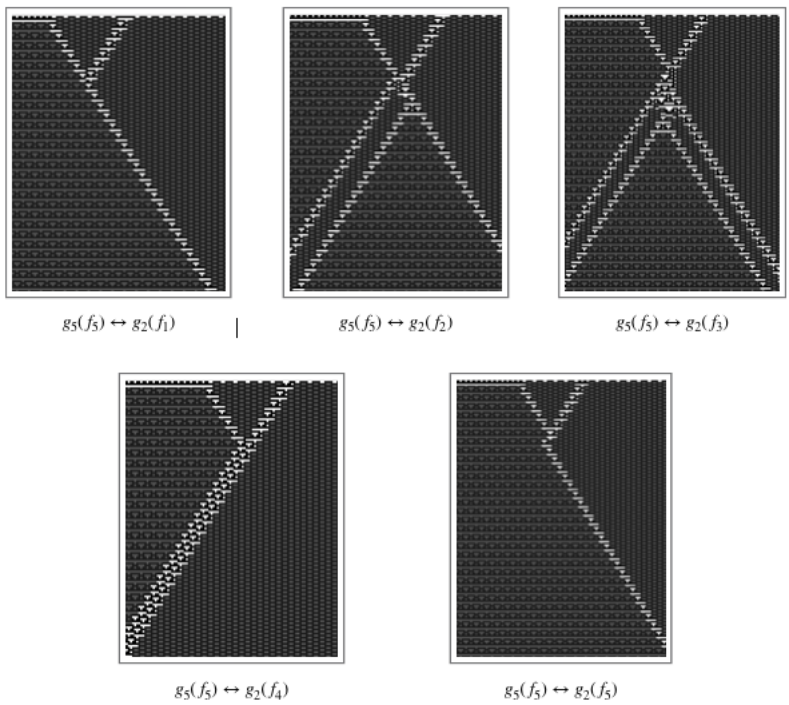


Figura B.30: The 5 different collisions between the glider g_5 in phase 5 and the 5 phases that the glider g_2 has

So the equations that we have as a result are:

1. $fp_2(f_5) \leftrightarrow g_5(f_5) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_1) \leftrightarrow fp_1(f_1) = g_5$

2. $fp_2(f_5) \leftrightarrow g_5(f_5) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_2) \leftrightarrow fp_1(f_2) = g_4 + g_6 + g_5$
3. $fp_2(f_5) \leftrightarrow g_5(f_5) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_3) \leftrightarrow fp_1(f_2) = g_4 + g_6 + g_5 + g_2$
4. $fp_2(f_5) \leftrightarrow g_5(f_5) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_4) \leftrightarrow fp_1(f_1) = g_4^2$
5. $fp_2(f_5) \leftrightarrow g_5(f_5) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_5) \leftrightarrow fp_1(f_2) = g_5$

B.1.9. g5 VS g6

To g_5 against g_6, we have both have 5 phases, so the number of collisions that we can perform are:

$$Collisions = 5 * 5 = 25 \tag{B.10}$$

g5 in the phase 1

In the figure B.31 we can observe the 5 possible collisions by keeping the glider g_5 phase fixed using phase 1, and rotate the phase of the glider g_6 in the 5 phases that it has.

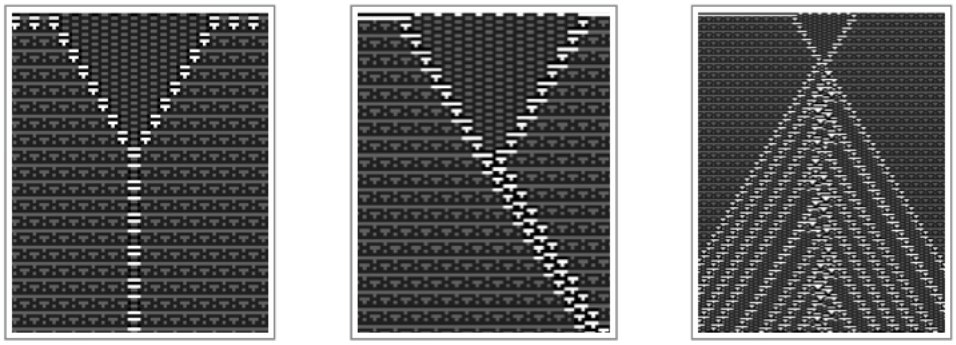
So the equations that we have as a result are:

1. $fp_2(f_1) \leftrightarrow g_5(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_1) \leftrightarrow fp_2(f_1) = s_2$
2. $fp_2(f_1) \leftrightarrow g_5(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_2) \leftrightarrow fp_2(f_2) = g_5 g_3$
3. $fp_2(f_1) \leftrightarrow g_5(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_3) \leftrightarrow fp_2(f_3) = gun_3$
4. $fp_2(f_1) \leftrightarrow g_5(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_4) \leftrightarrow fp_2(f_4) = gun_4$
5. $fp_2(f_1) \leftrightarrow g_5(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_5) \leftrightarrow fp_2(f_5) = g_4 g_6$

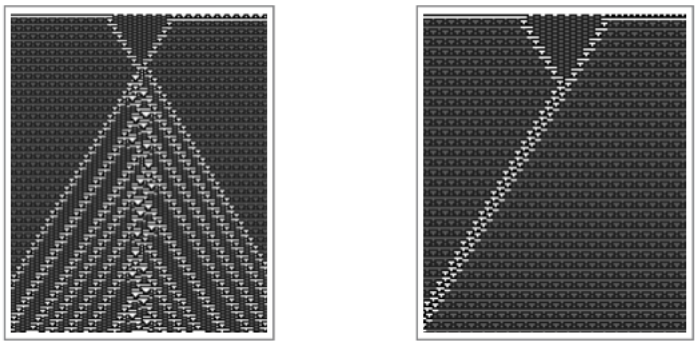
g5 in the phase 2

In the figure B.32 we can observe the 5 possible collisions by keeping the glider g_5 phase fixed using phase 2, and rotate the phase of the glider g_6 in the 5 phases that it has.

So the equations that we have as a result are:



$g_5(f_1) \leftrightarrow g_6(f_1)$ $g_5(f_1) \leftrightarrow g_6(f_2)$ $g_5(f_1) \leftrightarrow g_6(f_3)$



$g_5(f_1) \leftrightarrow g_6(f_4)$ $g_5(f_1) \leftrightarrow g_6(f_5)$

Figura B.31: The 5 different collisions between the glider g_5 in phase 1 and the 5 phases that the glider g_6 has

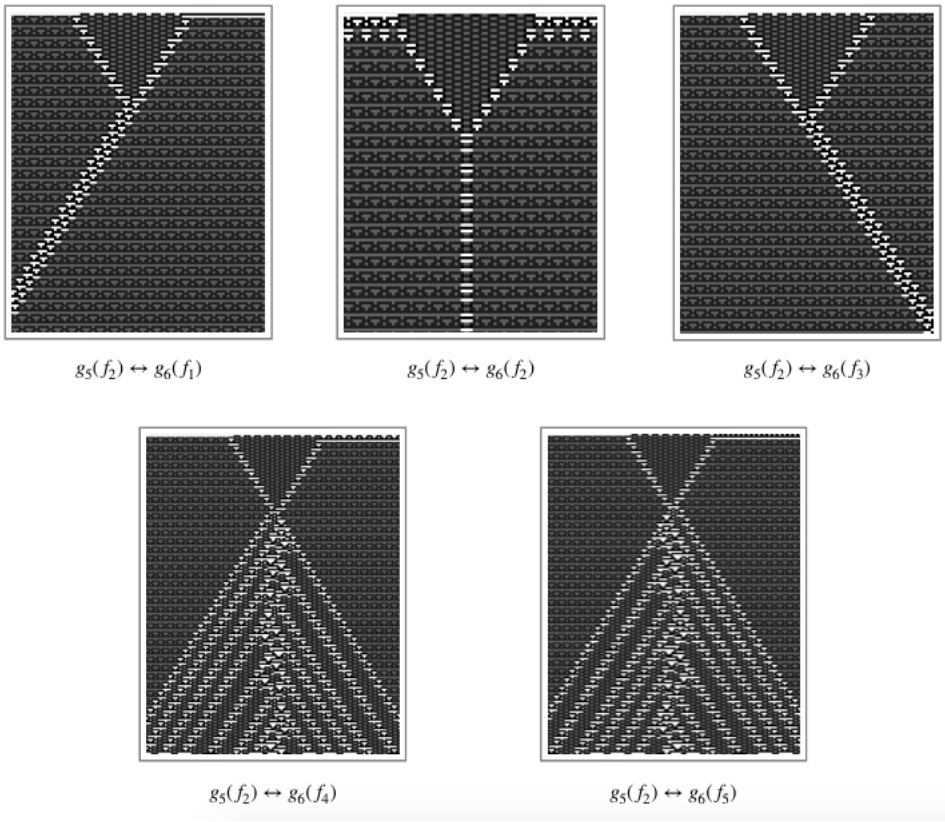


Figura B.32: The 5 different collisions between the glider g_5 in phase 2 and the 5 phases that the glider g_6 has

1. $fp_2(f_2) \leftrightarrow g_5(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_1) \leftrightarrow fp_2(f_1) = g_4 g_6$
2. $fp_2(f_2) \leftrightarrow g_5(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_2) \leftrightarrow fp_2(f_2) = s_2$
3. $fp_2(f_2) \leftrightarrow g_5(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_3) \leftrightarrow fp_2(f_3) = g_5 g_3$
4. $fp_2(f_2) \leftrightarrow g_5(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_4) \leftrightarrow fp_2(f_4) = gun_3$
5. $fp_2(f_2) \leftrightarrow g_5(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_5) \leftrightarrow fp_2(f_5) = gun_4$

g5 in the phase 3

In the figure B.33 we can observe the 5 possible collisions by keeping the glider g_5 phase fixed using phase 3, and rotate the phase of the glider g_6 in the 5 phases that it has.

So the equations that we have as a result are:

1. $fp_2(f_3) \leftrightarrow g_5(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_1) \leftrightarrow fp_2(f_1) = gun_4$
2. $fp_2(f_3) \leftrightarrow g_5(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_2) \leftrightarrow fp_2(f_2) = g_4 g_6$
3. $fp_2(f_3) \leftrightarrow g_5(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_3) \leftrightarrow fp_2(f_3) = s_2$
4. $fp_2(f_3) \leftrightarrow g_5(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_4) \leftrightarrow fp_2(f_4) = g_5 g_3$
5. $fp_2(f_3) \leftrightarrow g_5(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_5) \leftrightarrow fp_2(f_5) = gun_3$

g5 in the phase 4

In the figure B.34 we can observe the 5 possible collisions by keeping the glider g_5 phase fixed using phase 4, and rotate the phase of the glider g_6 in the 5 phases that it has.

So the equations that we have as a result are:

1. $fp_2(f_4) \leftrightarrow g_5(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_1) \leftrightarrow fp_2(f_1) = gun_3$
2. $fp_2(f_4) \leftrightarrow g_5(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_2) \leftrightarrow fp_2(f_2) = gun_4$
3. $fp_2(f_4) \leftrightarrow g_5(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_3) \leftrightarrow fp_2(f_3) = g_4 g_6$

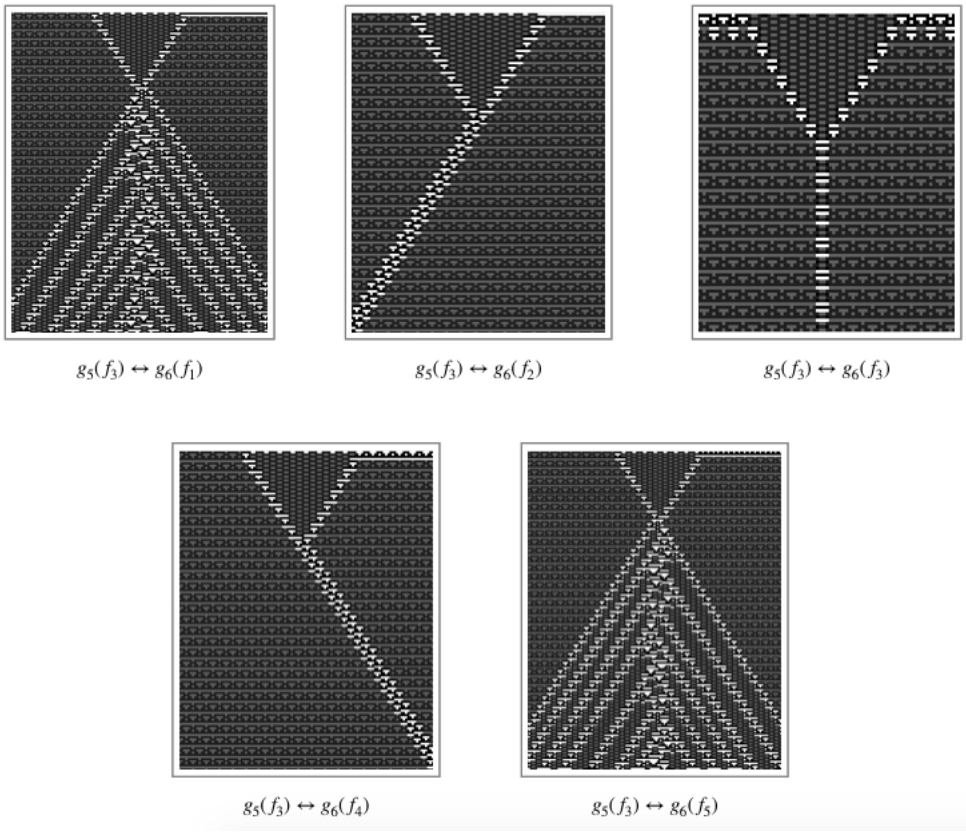


Figura B.33: The 5 different collisions between the glider g_5 in phase 3 and the 5 phases that the glider g_6 has

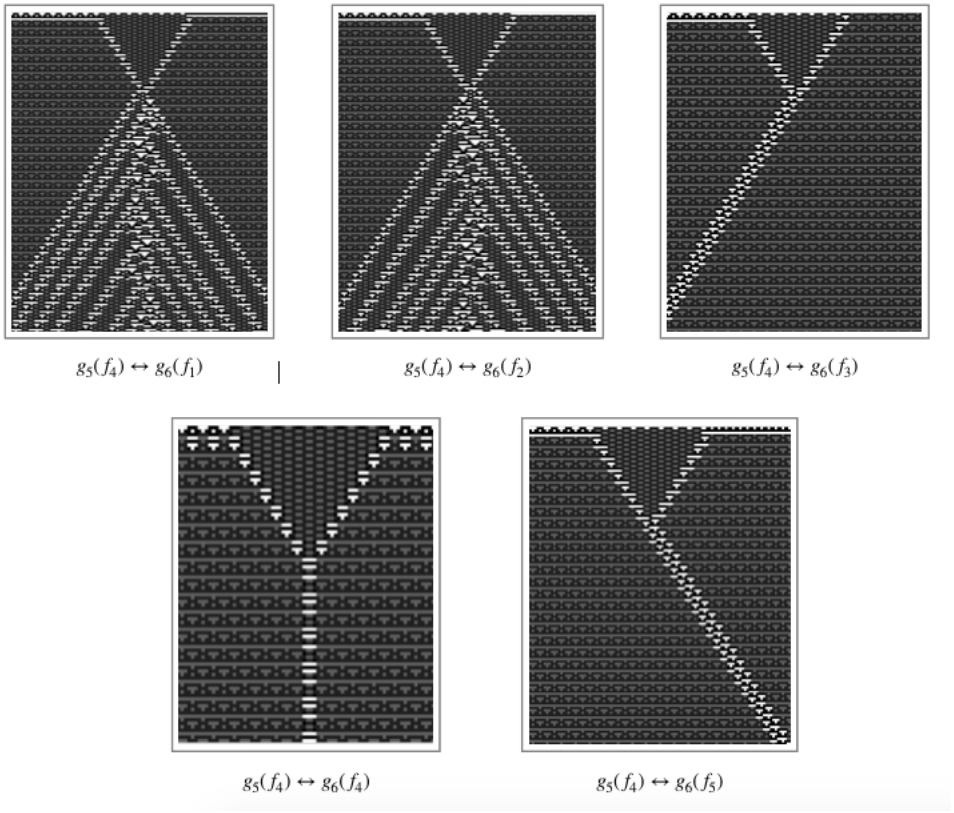


Figura B.34: The 5 different collisions between the glider g_5 in phase 4 and the 5 phases that the glider g_6 has

4. $fp_2(f_4) \leftrightarrow g_5(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_4) \leftrightarrow fp_2(f_4) = s_2$

5. $fp_2(f_4) \leftrightarrow g_5(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_5) \leftrightarrow fp_2(f_5) = g_5 g_3$

g5 in the phase 5

In the figure B.35 we can observe the 5 possible collisions by keeping the glider g_5 phase fixed using phase 5, and rotate the phase of the glider g_6 in the 5 phases that it has.

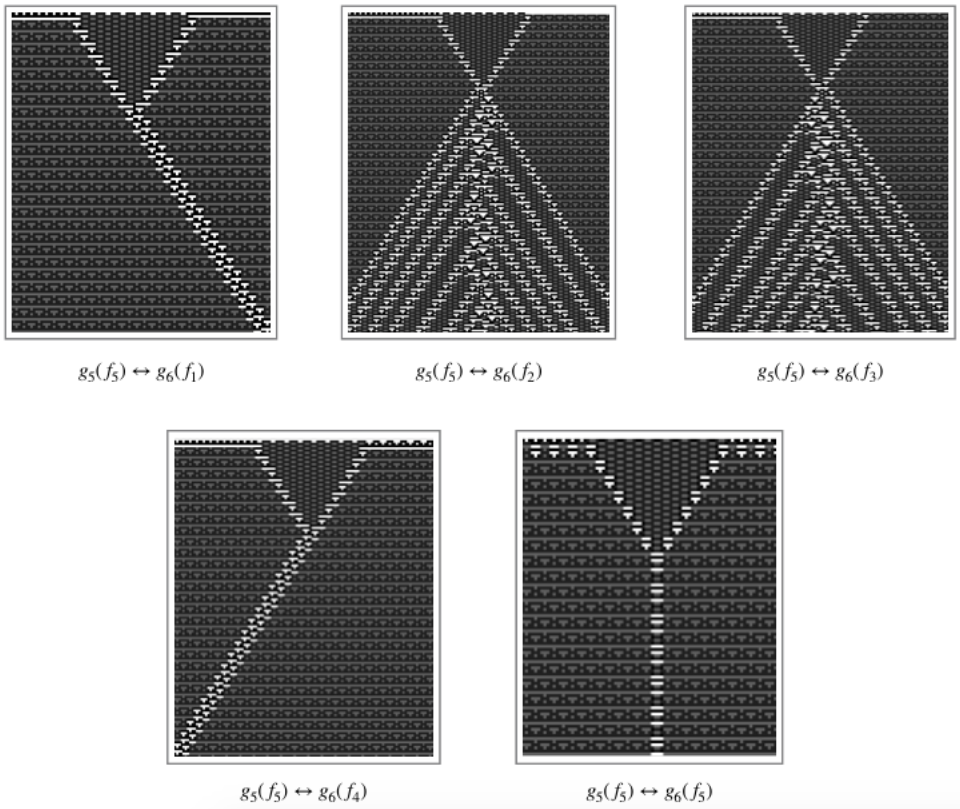


Figura B.35: The 5 different collisions between the glider g_5 in phase 5 and the 5 phases that the glider g_6 has

So the equations that we have as a result are:

1. $fp_2(f_5) \leftrightarrow g_5(f_5) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_1) \leftrightarrow fp_2(f_1) = g_5 g_3$

2. $fp_2(f_5) \leftrightarrow g_5(f_5) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_2) \leftrightarrow fp_2(f_2) = gun_3$

3. $fp_2(f_5) \leftrightarrow g_5(f_5) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_3) \leftrightarrow fp_2(f_3) = gun_4$
4. $fp_2(f_5) \leftrightarrow g_5(f_5) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_4) \leftrightarrow fp_2(f_4) = g_4 g_6$
5. $fp_2(f_5) \leftrightarrow g_5(f_5) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_5) \leftrightarrow fp_2(f_5) = s_2$

B.1.10. g5 VS s1

To g_5 against s_1, we have that the glider g_5 has 5 phases and the still-life s_1 has 2, so the number of collisions we can make are:

$$Collisions = 5 * 2 = 10 \tag{B.11}$$

g5 in the phase 1

In the figure B.36 we can observe the 5 possible collisions by keeping the glider g_5 phase fixed using phase 1, and rotate the still-life s_1 phase in the 2 phases that it has.

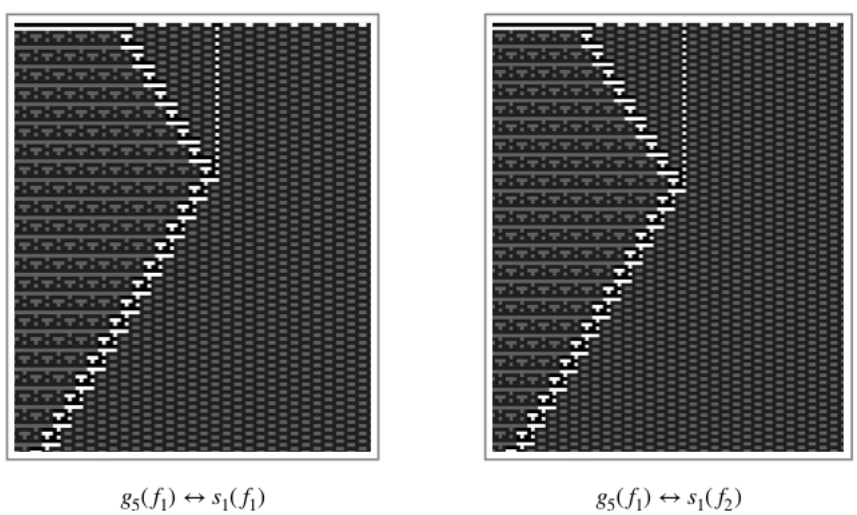


Figura B.36: The 5 different collisions between the glider g_5 in phase 1 and the 2 phases of the still-life s_1

So the equations that we have as a result are:

1. $fp_2(f_1) \leftrightarrow g_5(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_1) = g_4$
2. $fp_2(f_1) \leftrightarrow g_5(f_1) \leftrightarrow fp_1(f_1) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_2) = g_4$

g5 in the phase 2

In the figure B.37 we can observe the 5 possible collisions by keeping the glider g_5 phase fixed using phase 2, and rotate the still-life s_1 phase in the 2 phases that it has.

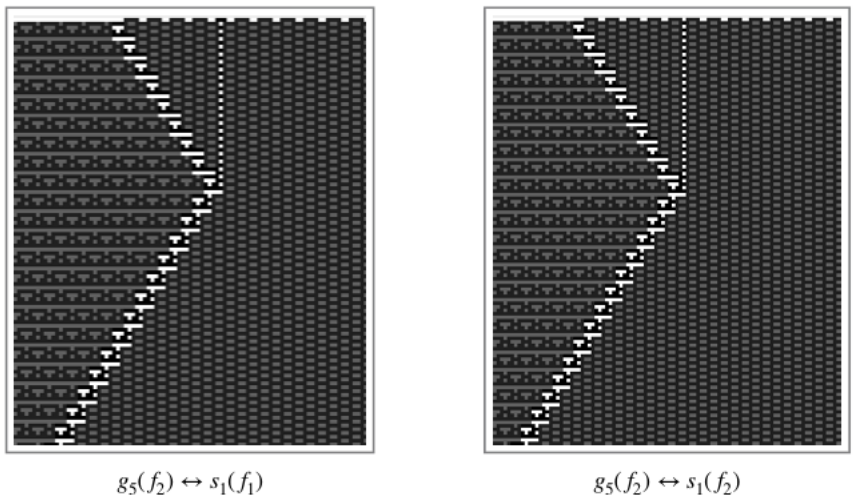


Figura B.37: The 5 different collisions between the glider g_5 in phase 2 and the 2 phases of the still-life s_1

So the equations that we have as a result are:

1. $fp_2(f_2) \leftrightarrow g_5(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_1) = g_4$
2. $fp_2(f_2) \leftrightarrow g_5(f_2) \leftrightarrow fp_1(f_2) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_2) = g_4$

g5 in the phase 3

In the figure B.38 we can observe the 5 possible collisions by keeping the glider g_5 phase fixed using phase 3, and rotate the still-life s_1 phase in the 2 phases that it has.

So the equations that we have as a result are:

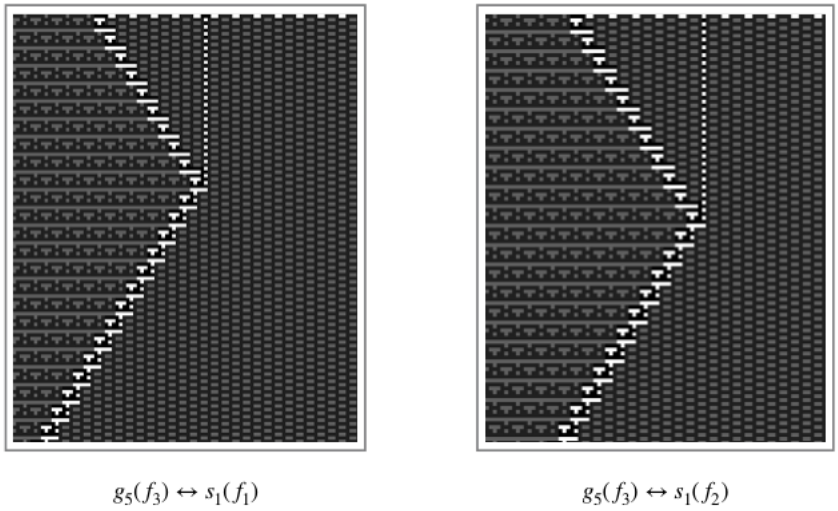


Figura B.38: The 5 different collisions between the glider g_5 in phase 3 and the 2 phases of the still-life s_1

1. $fp_2(f_3) \leftrightarrow g_5(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_1) = g_4$
2. $fp_2(f_3) \leftrightarrow g_5(f_3) \leftrightarrow fp_1(f_1) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_2) = g_4$

g5 in the phase 4

In the figure B.39 we can observe the 5 possible collisions by keeping the glider g_5 phase fixed using phase 4, and rotate the still-life s_1 phase in the 2 phases that it has.

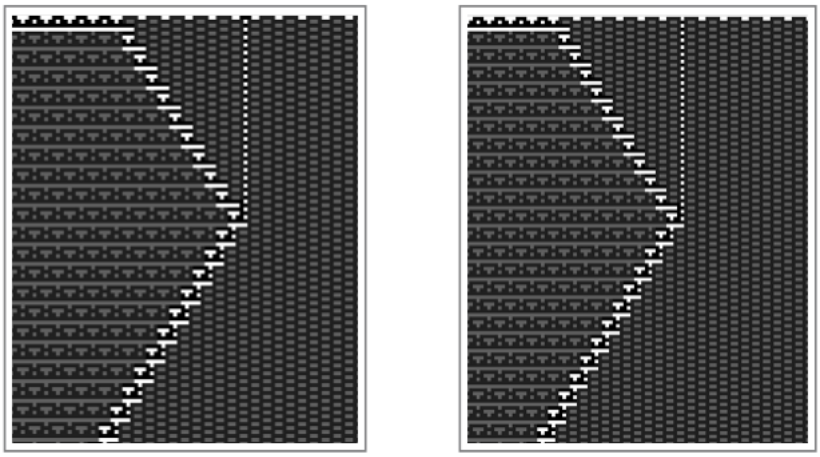
So the equations that we have as a result are:

1. $fp_2(f_4) \leftrightarrow g_5(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_1) = g_4$
2. $fp_2(f_4) \leftrightarrow g_5(f_4) \leftrightarrow fp_1(f_2) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_2) = g_4$

g5 in the phase 5

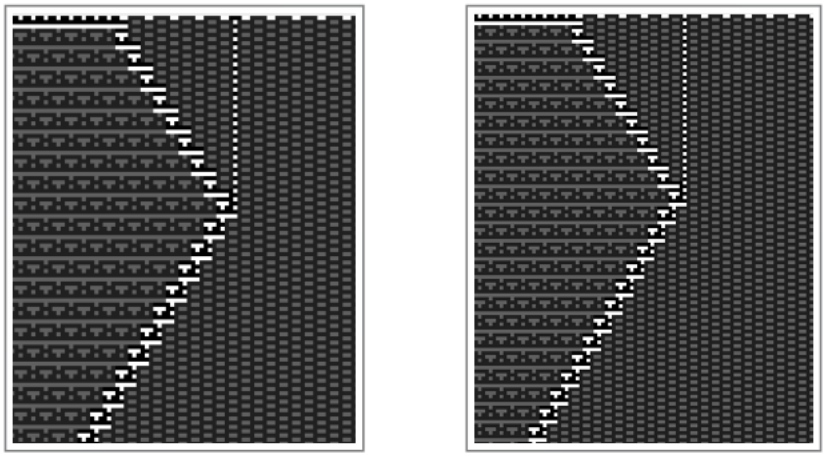
In the figure B.40 we can observe the 5 possible collisions by keeping the glider g_5 phase fixed using phase 5, and rotate the still-life s_1 phase in the 2 phases that it has.

So the equations that we have as a result are:



$g_5(f_4) \leftrightarrow s_1(f_1)$ $g_5(f_4) \leftrightarrow s_1(f_2)$

Figura B.39: The 5 different collisions between the glider g_5 in phase 4 and the 2 phases of the still-life s_1



$g_5(f_5) \leftrightarrow s_1(f_1)$ $g_5(f_5) \leftrightarrow s_1(f_2)$

Figura B.40: The 5 different collisions between the glider g_5 in phase 5 and the 2 phases of the still-life s_1

1. $fp_2(f_5) \leftrightarrow g_5(f_5) \leftrightarrow fp_1(f_1) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_1) = g_4$
2. $fp_2(f_5) \leftrightarrow g_5(f_5) \leftrightarrow fp_1(f_1) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_2) = g_4$

B.1.11. s1 VS g2

To s_1 against g_2, we have that the still-life s_1 has 2 phases and the glider g_2 has 5, so the number of collisions we can make are:

$$Collisions = 2 * 5 = 10 \tag{B.12}$$

s1 In the phase 1

In the figure B.41 we can observe the 5 possible collisions by keeping the still-life s_1 phase fixed using phase 1, and rotate the phase of the glider g_2 in the 5 phases that it has.

So the equations that we have as a result are:

1. $fp_1(f_2) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_1) \leftrightarrow fp_1(f_1) = s_1$
2. $fp_1(f_2) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_2) \leftrightarrow fp_1(f_2) = s_1$
3. $fp_1(f_2) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_3) \leftrightarrow fp_1(f_2) = s_1$
4. $fp_1(f_2) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_4) \leftrightarrow fp_1(f_1) = s_1$
5. $fp_1(f_2) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_2) \leftrightarrow g_2(f_5) \leftrightarrow fp_1(f_2) = s_1$

s1 in the phase 2

In the figure B.42 we can observe the 5 possible collisions by keeping the still-life s_1 phase fixed using phase 2, and rotate the phase of the glider g_2 in the 5 phases that it has.

So the equations that we have as a result are:

1. $fp_1(f_1) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_1) \leftrightarrow fp_1(f_1) = s_1$

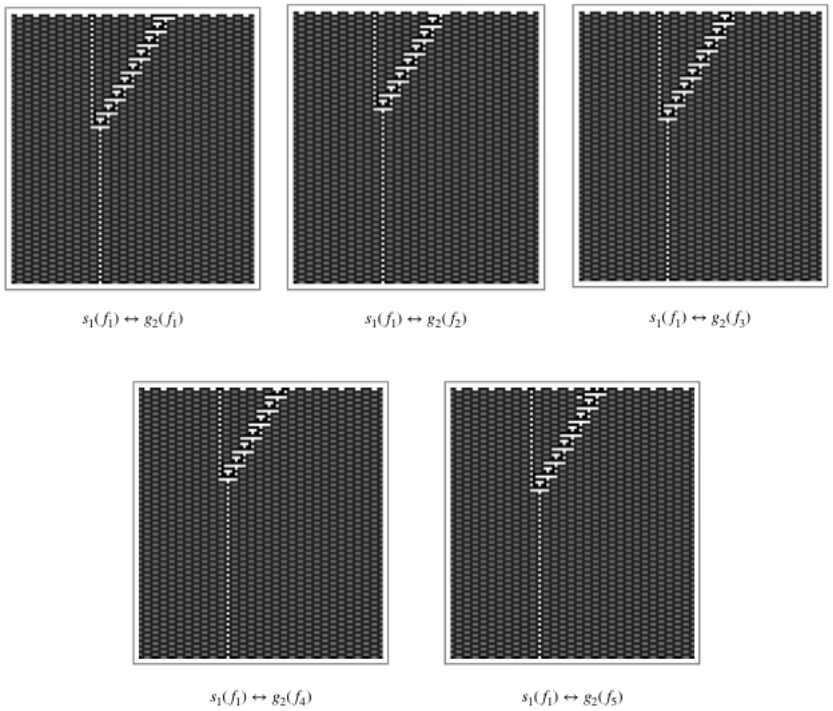


Figura B.41: The 5 different collisions between the still-life s_1 in phase 1 and the 5 phases that the glider g_2 has

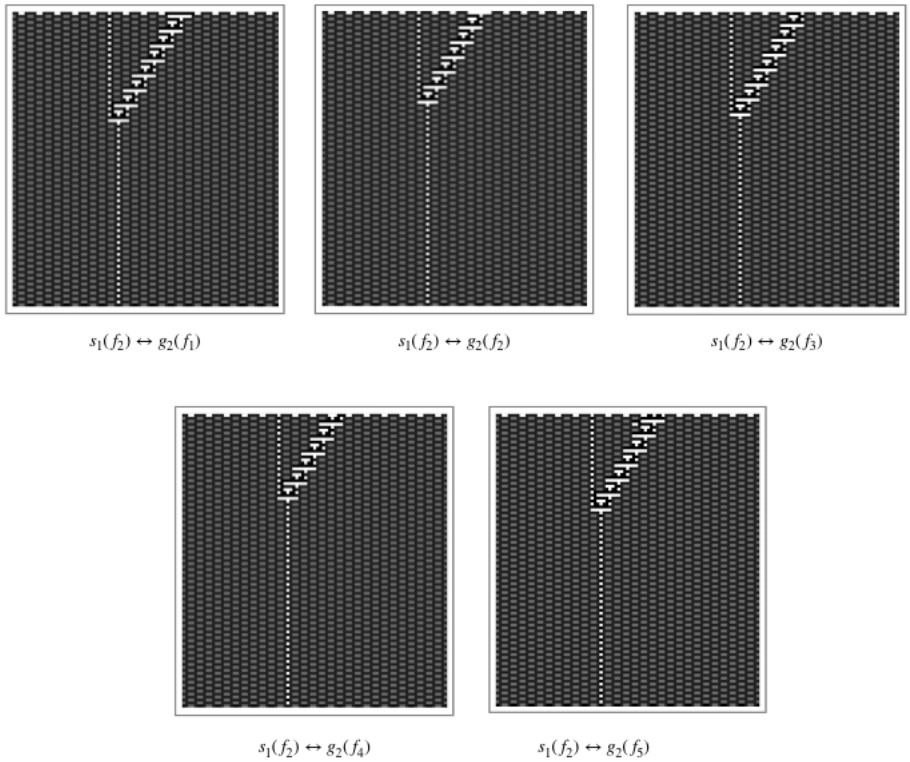


Figura B.42: The 5 different collisions between the still-life s_1 in phase 2 and the 5 phases that the glider g_2 has

2. $fp_1(f_1) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_2) \leftrightarrow fp_1(f_2) = s_1$
3. $fp_1(f_1) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_3) \leftrightarrow fp_1(f_2) = s_1$
4. $fp_1(f_1) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_4) \leftrightarrow fp_1(f_1) = s_1$
5. $fp_1(f_1) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_1) \leftrightarrow g_2(f_5) \leftrightarrow fp_1(f_2) = s_1$

B.1.12. s1 VS g6

To s_1 against g_6, we have the still-life s_1 has 2 phases and the glider g_6 has 5, so the number of collisions we can make are:

$$Collisions = 2 * 5 = 10 \tag{B.13}$$

s1 in the phase 1

In the figure B.43 we can observe the 5 possible collisions by keeping the still-life s_1 phase fixed using phase 1, and rotate the phase of the glider g_6 in the 5 phases that it has.

So the equations that we have as a result are:

1. $fp_1(f_2) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_1) \leftrightarrow fp_2(f_1) = g_3$
2. $fp_1(f_2) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_2) \leftrightarrow fp_2(f_2) = g_3$
3. $fp_1(f_2) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_3) \leftrightarrow fp_2(f_3) = g_3$
4. $fp_1(f_2) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_4) \leftrightarrow fp_2(f_4) = g_3$
5. $fp_1(f_2) \leftrightarrow s_1(f_1) \leftrightarrow fp_1(f_2) \leftrightarrow g_6(f_5) \leftrightarrow fp_2(f_5) = g_3$

s1 in the phase 2

In the figure B.44 we can observe the 5 possible collisions by keeping the still-life s_1 phase fixed using phase 2, and rotate the phase of the glider g_6 in the 5 phases that it has.

So the equations that we have as a result are:

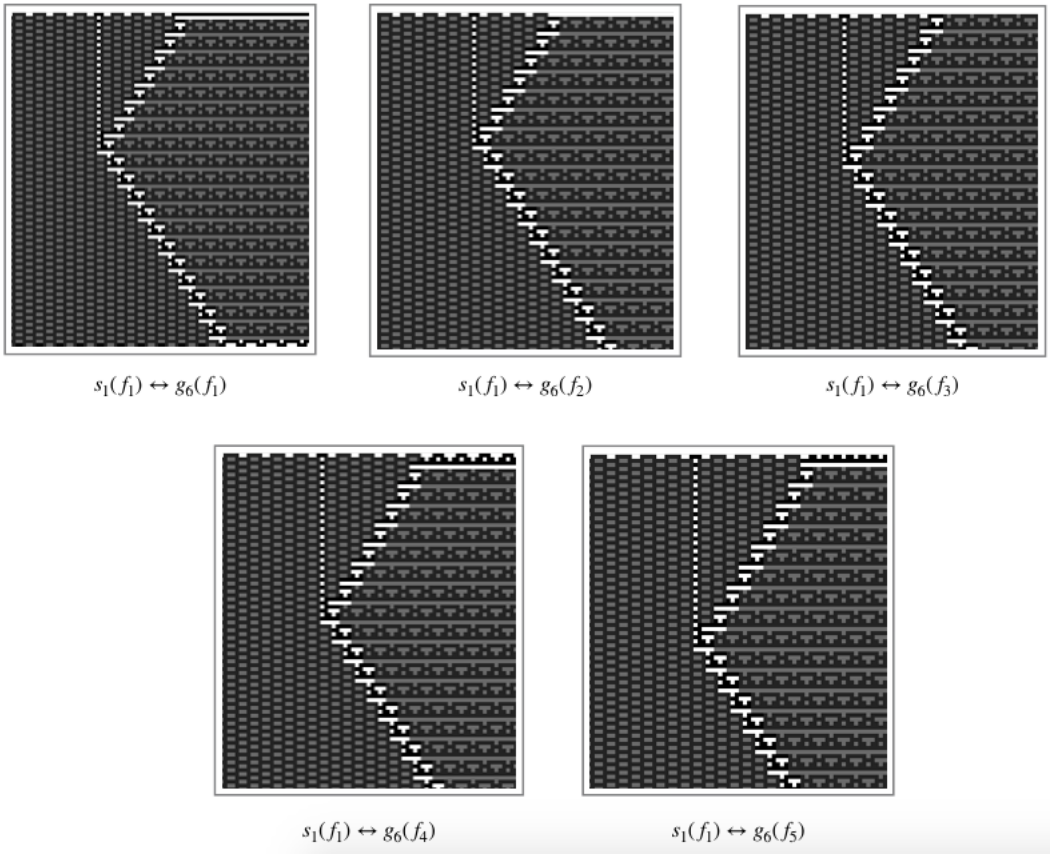


Figura B.43: The 5 different collisions between the still-life s_1 in phase 1 and the 5 phases that the glider g_6 has

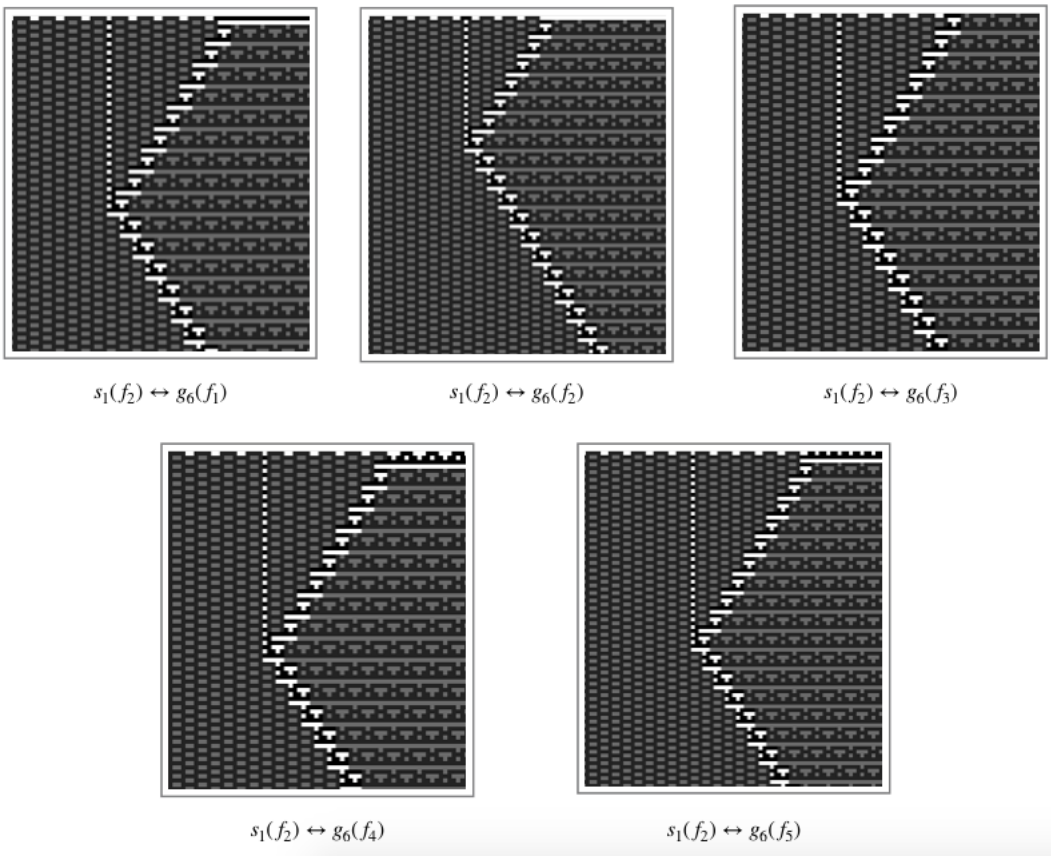


Figura B.44: The 5 different collisions between the still-life s_1 in phase 2 and the 5 phases that the glider g_6 has

1. $fp_1(f_1) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_1) \leftrightarrow fp_2(f_1) = g_3$
2. $fp_1(f_1) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_2) \leftrightarrow fp_2(f_2) = g_3$
3. $fp_1(f_1) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_3) \leftrightarrow fp_2(f_3) = g_3$
4. $fp_1(f_1) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_4) \leftrightarrow fp_2(f_4) = g_3$
5. $fp_1(f_1) \leftrightarrow s_1(f_2) \leftrightarrow fp_1(f_1) \leftrightarrow g_6(f_5) \leftrightarrow fp_2(f_5) = g_3$

B.1.13. s2 VS g4

To s_2 against g_4, we have the still-life s_2 has 10 phases and the glider g_4 has 5, but we need to apply the restriction number 1 mentioned in B.1.2 to the glider g_4, so we can only collide at one point, obtaining as totals:

$$Collisions = 10 * 1 = 10 \tag{B.14}$$

s2 in the phase 1

In the figure B.45 we can observe the possible collision by keeping the still-life s_2 phase fixed using phase 1, and the glider g_4 in phase 1.

So the equation that we have as a result is:

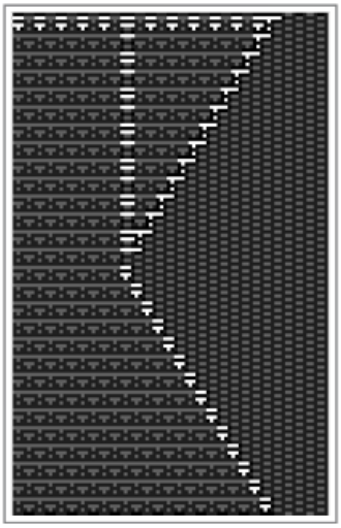
1. $fp_2(f_1) \leftrightarrow s_2(f_1) \leftrightarrow fp_2(f_1) \leftrightarrow g_4(f_1) \leftrightarrow fp_1(f_2) = g_5$

s2 in the phase 2

In the figure B.46 we can observe the possible collision by keeping the still-life s_2 phase fixed using phase 2, and the glider g_4 in phase 2.

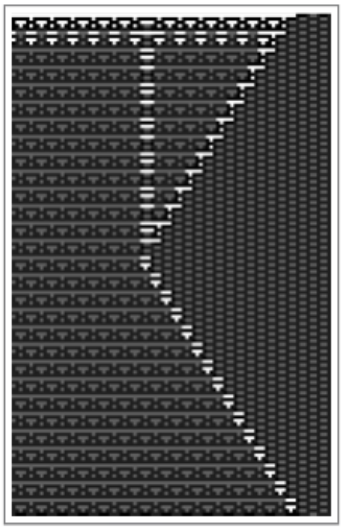
So the equation that we have as a result is:

1. $fp_2(f_2) \leftrightarrow s_2(f_2) \leftrightarrow fp_2(f_2) \leftrightarrow g_4(f_2) \leftrightarrow fp_1(f_1) = g_5$



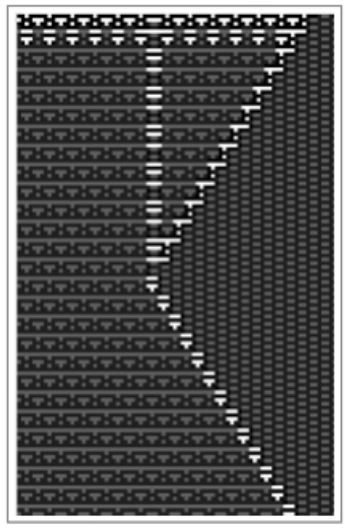
$s_2(f_1) \leftrightarrow g_4(p_1)$

Figura B.45: The collision between the still-life s_2 in phase 1 and the glider g_4 in phase 1



$s_2(f_2) \leftrightarrow g_4(p_2)$

Figura B.46: The collision between the still-life s_2 in phase 2 and the glider g_4 in phase 2



$s_2(f_3) \leftrightarrow g_4(p_3)$

Figura B.47: The collision between the still-life s_2 in phase 3 and the glider g_4 in phase 3

s2 in the phase 3

In the figure B.47 we can observe the possible collision by keeping the still-life s_2 phase fixed using phase 3, and the glider g_4 in phase 3.

So the equation that we have as a result is:

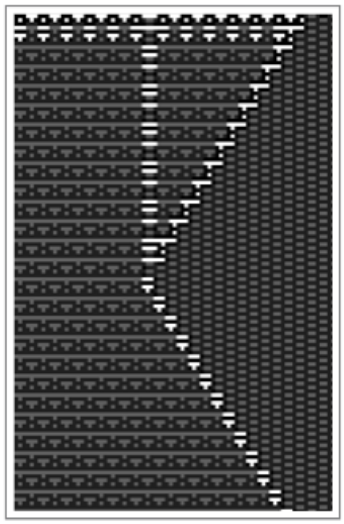
1. $fp_2(f_3) \leftrightarrow s_2(f_3) \leftrightarrow fp_2(f_3) \leftrightarrow g_4(f_3) \leftrightarrow fp_1(f_2) = g_5$

s2 in the phase 4

In the figure B.48 we can observe the possible collision by keeping the still-life s_2 phase fixed using phase 4, and the glider g_4 in phase 4.

So the equation that we have as a result is:

1. $fp_2(f_4) \leftrightarrow s_2(f_4) \leftrightarrow fp_2(f_4) \leftrightarrow g_4(f_4) \leftrightarrow fp_1(f_1) = g_5$



$s_2(f_4) \leftrightarrow g_4(p_4)$

Figura B.48: The collision between the still-life s_2 in phase 4 and the glider g_4 in phase 4

s2 in the phase 5

In the figure B.49 we can observe the possible collision by keeping the still-life s_2 phase fixed using phase 5, and the glider g_4 in phase 5.

So the equation that we have as a result is:

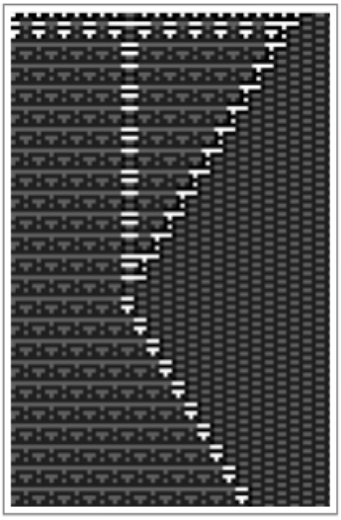
1. $fp_2(f_5) \leftrightarrow s_2(f_5) \leftrightarrow fp_2(f_5) \leftrightarrow g_4(f_5) \leftrightarrow fp_1(f_2) = g_5$

s2 in the phase 6

In the figure B.50 we can observe the possible collision by keeping the still-life s_2 phase fixed using phase 6, and the glider g_4 in phase 1.

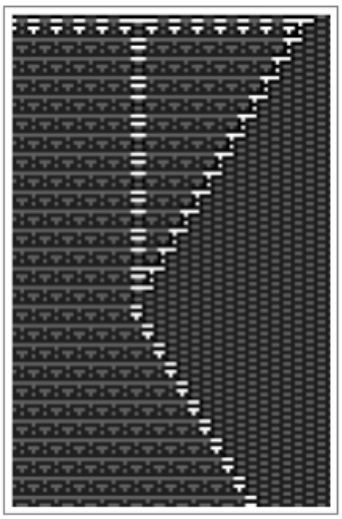
So the equation that we have as a result is:

1. $fp_2(f_1) \leftrightarrow s_2(f_6) \leftrightarrow fp_2(f_1) \leftrightarrow g_4(f_1) \leftrightarrow fp_1(f_1) = g_5$



$s_2(f_5) \leftrightarrow g_4(p_5)$

Figura B.49: The collision between the still-life s_2 in phase 5 and the glider g_4 in phase 5

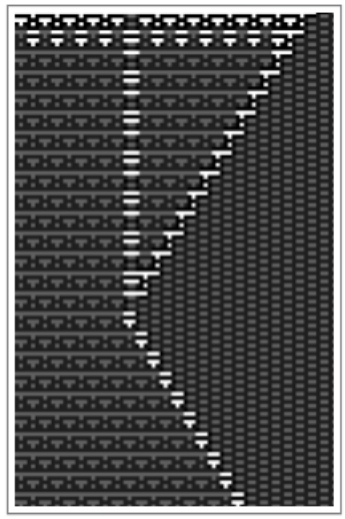


$s_2(f_6) \leftrightarrow g_4(p_1)$

Figura B.50: The collision between the still-life s_2 in phase 6 and the glider g_4 in phase 1

s2 in the phase 7

In the figure B.51 we can observe the possible collision by keeping the still-life s_2 phase fixed using phase 7, and the glider g_4 in phase 2.



$s_2(f_7) \leftrightarrow g_4(p_2)$

Figura B.51: The collision between the still-life s_2 in phase 7 and the glider g_4 in phase 2

So the equation that we have as a result is:

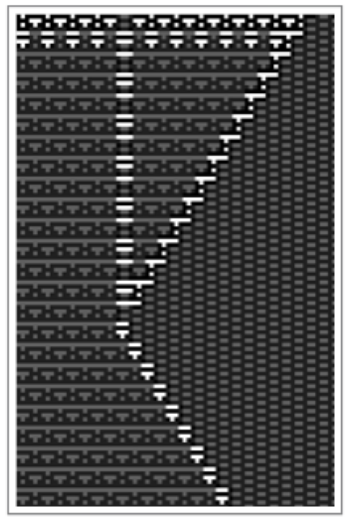
1. $fp_2(f_2) \leftrightarrow s_2(f_7) \leftrightarrow fp_2(f_2) \leftrightarrow g_4(f_2) \leftrightarrow fp_1(f_2) = g_5$

s2 in the phase 8

In the figure B.52 we can observe the possible collision by keeping the still-life s_2 phase fixed using phase 8, and the glider g_4 in phase 3.

So the equation that we have as a result is:

1. $fp_2(f_3) \leftrightarrow s_2(f_8) \leftrightarrow fp_2(f_3) \leftrightarrow g_4(f_3) \leftrightarrow fp_1(f_2) = g_5$



$s_2(f_8) \leftrightarrow g_4(p_3)$

Figura B.52: The collision between the still-life s_2 in phase 8 and the glider g_4 in phase 3

s2 in the phase 9

In the figure B.53 we can observe the possible collision by keeping the still-life s_2 phase fixed using phase 9, and the glider g_4 in phase 4.

So the equation that we have as a result is:

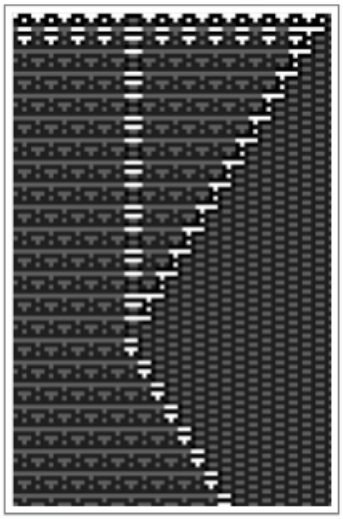
1. $fp_2(f_4) \leftrightarrow s_2(f_9) \leftrightarrow fp_2(f_4) \leftrightarrow g_4(f_4) \leftrightarrow fp_1(f_1) = g_5$

s2 in the phase 10

In the figure B.54 we can observe the possible collision by keeping the still-life s_2 phase fixed using phase 10, and the glider g_4 in phase 5.

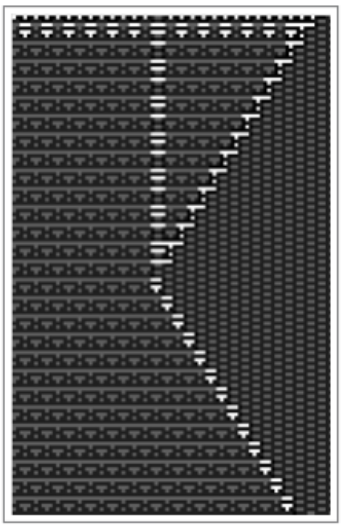
So the equation that we have as a result is:

1. $fp_2(f_5) \leftrightarrow s_2(f_{10}) \leftrightarrow fp_2(f_5) \leftrightarrow g_4(f_5) \leftrightarrow fp_1(f_2) = g_5$



$s_2(f_9) \leftrightarrow g_4(p_4)$

Figura B.53: The collision between the still-life s_2 in phase 9 and the glider g_4 in phase 4



$s_2(f_{10}) \leftrightarrow g_4(p_5)$

Figura B.54: The collision between the still-life s_2 in phase 10 and the glider g_4 in phase 5