



Instituto Politécnico Nacional
Escuela Superior de Cómputo

Trabajo Terminal

**EXPLORACIÓN DE UNIVERSOS DISCRETOS CON AUTÓMATAS
CELULARES EN DOS DIMENSIONES**

No. de Registro: 2019-B067

Para optar al grado de
“Ingeniería en Sistemas Computacionales”

Presentan
Esquivel Valdez Alberto
Torres Hernández Eduardo
Vargas Romero Erick Efraín

Director
Dr. Genaro Juárez Martínez

Ciudad de México a 24 de febrero de 2021

Índice general

I	Introducción	11
II	Estado del arte	13
1	Sistemas dinámicos	14
1.1	Modelo continuo y discreto	14
1.1.1	Sistemas dinámicos a tiempo continuo	14
1.1.2	Sistemas dinámicos a tiempo discreto	15
1.2	Teoría del caos	15
1.2.1	Determinismo	15
1.2.2	Acotado	15
1.2.3	Irregularidad	16
1.2.4	Sensible a las condiciones iniciales	16
1.2.5	Imprevisibilidad	17
1.2.6	Atractores caóticos	17
1.3	Sistemas complejos	19
1.3.1	Complejidad	19
1.3.2	No linealidad	19
1.3.3	Retroalimentación	19
1.3.4	Orden espontáneo	19
1.3.5	Robustez y falta de control central	20
1.3.6	Emergencia	20
1.3.7	Organización jerárquica	20
2	Autómatas Celulares	21
2.1	Antecedentes históricos	21
2.2	Conceptos básicos	21
2.3	AC en 1D, 2D y 3D	22
2.3.1	AC en una dimensión	22
2.3.2	AC en dos dimensiones	23
2.3.3	Vecindades	23
2.3.4	Tipos de reglas	24
2.3.5	AC en tres dimensiones	26

2.4	Autómata Celular complejo	27
2.4.1	Juego de la Vida (Game of Life)	27
2.5	Aplicaciones	28
2.5.1	Arquitectura	28
2.5.2	Criptografía	29
2.5.3	Modelado epidémico	30
2.6	Teoría del campo promedio	31
2.7	Clases de puntos fijos	32
2.8	Entropía	33
2.9	Sistemas similares	35
2.9.1	Golly	35
2.9.2	ACOSXL21, ACOSXSTV, ACOSXSTM, ACOSXV y ACOSXM	37
2.9.3	samcodes.co	43
III	Análisis y Diseño	45
3	Introducción a la metodología	46
4	Fase 1: Planificación de requerimientos	47
4.1	Problemática	47
4.1.1	Planteamiento de solución	47
4.2	Requerimientos	47
4.2.1	Requerimientos de hardware	47
4.2.2	Requerimientos de software	48
4.2.3	Requerimientos funcionales	48
4.2.4	Requerimientos no funcionales	49
4.3	Herramientas utilizadas para el desarrollo	50
4.3.1	C++	50
4.3.2	Qt Creator	50
4.4	Arquitectura	50
4.4.1	Diagrama de clases	50
4.4.2	Diagrama BPMN	55
4.5	Casos de uso	77
4.5.1	CU01 Definir tamaño del espacio	77
4.5.2	CU02 Definir tipo de vecindad	78
4.5.3	CU03 Definir tipo de regla	79
4.5.4	CU04 Definir los valores de la regla totalista	79
4.5.5	CU05 Restablecer valores de la regla totalista	81
4.5.6	CU06 Definir los valores de la regla semitotalista	81
4.5.7	CU07 Definir tipo de espacio	83
4.5.8	CU08 Establecer configuración inicial del AC	84
4.5.9	CU09 Exportar el objeto “Cellular automata”	85
4.5.10	CU10 Definir tamaño de las células	87
4.5.11	CU11 Definir la velocidad de evolución del AC	88
4.5.12	CU12 Definir el escalar de proyección	89
4.5.13	CU13 Establecer el estado del gradiente	90
4.5.14	CU14 Establecer el estado de la rejilla	91
4.5.15	CU15 Definir el color de la rejilla	91
4.5.16	CU16 Definir color de las células	93
4.5.17	CU17 Cambiar el estado de una célula	95
4.5.18	CU18 Desplegar ventana de configuración	96
4.5.19	CU19 Desplegar el AC con la configuración inicial	97

4.5.20 CU20 Calcular siguiente generación	97
4.5.21 CU21 Desplegar siguiente generación	99
4.5.22 CU22 Calcular densidad de población	99
4.5.23 CU23 Calcular entropía	100
4.5.24 CU24 Calcular polinomio característico	100
4.5.25 CU25 Calcular puntos de intersección	101
4.5.26 CU26 Graficar densidad de población	103
4.5.27 CU27 Graficar entropía	103
4.5.28 CU28 Graficar polinomio característico	104
4.5.29 CU29 Graficar recta identidad	105
4.5.30 CU30 Graficar puntos de intersección	105
4.5.31 CU31 Iniciar simulación	106
4.5.32 CU32 Detener simulación	107
4.5.33 CU33 Restablecer simulación	108
4.6 Reglas de negocio	110
4.7 Catálogo de mensajes	114
5 Fase 2: Interfaz del usuario	115
5.1 Prototipos	115
5.1.1 Prototipo 01	115
5.1.2 Prototipo 02	118
5.1.3 Prototipo 03	121
5.1.4 Prototipo 04	123
6 Fase 3: Desarrollo	131
6.1 Pantallas	131
6.1.1 IU-CU01: Definir la altura del espacio	131
6.1.2 IU-CU01.1: Definir la anchura del espacio	132
6.1.3 IU-CU02: Definir el tipo de vecindad como Moore	133
6.1.4 IU-CU02.1: Definir el tipo de vecindad como von Neumann	134
6.1.5 IU-CU03: Definir el tipo de regla como totalista	135
6.1.6 IU-CU04: Cambiar el valor de la célula para una determinada configuración	136
6.1.7 IU-CU04.1: Cambiar la densidad para una regla totalista aleatoria	137
6.1.8 IU-CU04.1-1: Establecer regla aleatoria	138
6.1.9 IU-CU04.2: Ingresar ID de la regla totalista	139
6.1.10 IU-CU05 Restablacer valores de la regla totalista	140
6.1.11 IU-CU06: Definir N_{min}	141
6.1.12 IU-CU06.1: Definir N_{max}	142
6.1.13 IU-CU06.2: Definir S_{min}	143
6.1.14 IU-CU06.3: Definir S_{max}	144
6.1.15 IU-CU07: Definir tipo de espacio como cerrado	145
6.1.16 IU-CU07.1: Definir tipo de espacio como abierto	146
6.1.17 IU-CU08: Establecer configuración inicial aleatoria	147
6.1.18 IU-CU08.1: Establecer configuración inicial desde archivo	148
6.1.19 IU-CU08.1-1: Importar archivo	149
6.1.20 IU-CU08.1-2: Ventana de selección de archivo	150
6.1.21 IU-CU09: Exportar objeto AC	151
6.1.22 IU-CU09.1: Seleccionar directorio	152
6.1.23 IU-CU10: Definir el tamaño de las células	153
6.1.24 IU-CU11: Definir la velocidad de evolución del AC	154
6.1.25 IU-CU12: Asignar valor del escalar	155
6.1.26 IU-CU13: Estado del gradiente	156
6.1.27 IU-CU14: Estado de la rejilla	157

6.1.28	IU-CU14-1: Espacio de evolución del AC sin rejilla	158
6.1.29	IU-CU15: Definir el color de la rejilla	159
6.1.30	IU-CU15.1: Etiqueta indicadora de la rejilla	160
6.1.31	IU-CU15-1: Paleta de colores	161
6.1.32	IU-CU15.2: Rejilla en el espacio de evolución del AC	162
6.1.33	IU-CU16: Definir el color de las células vivas	163
6.1.34	IU-CU16.1: Definir el color de las células muertas	164
6.1.35	IU-CU16.2: Definir el color de las células de proyección	165
6.1.36	IU-CU16.3: Etiqueta indicadora de las células	166
6.1.37	IU-CU16-1: Células vivas en el espacio de evolución del AC	167
6.1.38	IU-CU16-2: Células muertas en el espacio de evolución del AC	168
6.1.39	IU-CU16-3: Células de proyección en el espacio de evolución del AC	169
6.1.40	IU-CU17: Cambiar una célula del espacio de evolución del AC	170
6.1.41	IU-CU18: Ventana de configuración	171
6.1.42	IU-CU19: Desplegar CA con la configuración inicial	172
6.1.43	IU-CU19.1: Espacio de evolución con la configuración inicial	173
6.1.44	IU-CU21: Desplegar siguiente generación	174
6.1.45	IU-CU21-1: Siguiente generación del AC	175
6.1.46	IU-CU26: Graficar población	176
6.1.47	IU-CU26.1: Gráfica de población	177
6.1.48	IU-CU27: Graficar entropía	178
6.1.49	IU-CU27.1: Gráfica de entropía	179
6.1.50	IU-CU28: Graficar polinomio característico	180
6.1.51	IU-CU28.1: Lista de puntos clasificados	181
6.1.52	IU-CU28.2: Gráfica del polinomio característico	182
6.1.53	IU-CU29: Graficar recta identidad	183
6.1.54	IU-CU29.1: Gráfica del polinomio característico con recta identidad	184
6.1.55	IU-CU30: Graficar puntos de intersección	185
6.1.56	IU-CU30.1: Gráfica del polinomio característico con puntos de intersección del tipo atractivo	186
6.1.57	IU-CU30.2: Gráfica del polinomio característico con puntos de intersección del tipo repulsivo	187
6.1.58	IU-CU30.3: Gráfica del polinomio característico con puntos de intersección del tipo crítico	188
6.1.59	IU-CU31: Iniciar simulación	189
6.1.60	IU-CU32: Detener simulación	190
6.1.61	IU-CU33: Restablecer simulación	191
6.1.62	IU-CU33.1: Restablecer gráfica de población	192
6.1.63	IU-CU33.2: Restablecer gráfica de entropía	193
6.1.64	IU-CU33.3: Restablecer espacio de evolución del AC	194
7	Fase 4: Implementación	196
7.1	Pruebas unitarias	196
7.1.1	PU01 Definir la altura del espacio	196
7.1.2	PU02 Definir la anchura del espacio	196
7.1.3	PU03 Definir vecindad del tipo Moore	197
7.1.4	PU04 Definir vecindad del tipo von Neumann	197
7.1.5	PU05 Definir el tipo de regla como totalista	198
7.1.6	PU06 Definir el tipo de regla como semitotalista	198
7.1.7	PU07 Definir los valores de la regla totalista	198
7.1.8	PU08 Restablecer valores de la regla totalista	199
7.1.9	PU09 Definir el valor de la variable N_{min}	199
7.1.10	PU10 Definir el valor de la variable N_{max}	200

7.1.11 PU11 Definir el valor de la variable S_{min}	200
7.1.12 PU08 Definir el valor de la variable S_{max}	201
7.1.13 PU13 Definir el tipo de espacio de evolución como cerrado	201
7.1.14 PU14 Definir el tipo de espacio de evolución como abierto	202
7.1.15 PU15 Establecer configuración inicial del AC de forma aleatoria	202
7.1.16 PU16 Establecer configuración inicial del AC desde archivo	202
7.1.17 PU17 Exportar objeto CA	203
7.1.18 PU18 Definir el tamaño de las células	203
7.1.19 PU19 Definir velocidad de evolución	204
7.1.20 PU20 Definir escalar de proyección	204
7.1.21 PU21 Establecer el estado del gradiente	205
7.1.22 PU22 Desactivar el gradiente	205
7.1.23 PU23 Establecer el estado de la rejilla	205
7.1.24 PU24 El valor actual de la rejilla es verdadero	206
7.1.25 PU25 Definir el color de la rejilla	206
7.1.26 PU26 Definir el color de las células	207
7.1.27 PU27 Definir el color de las células muertas	207
7.1.28 PU28 Definir el color de las células de proyección	207
7.1.29 PU29 Cambiar el estado de una célula	208
7.1.30 PU30 Desplegar ventana de configuración	208
7.1.31 PU31 Desplegar el AC con la configuración inicial	209
7.1.32 PU32 Calcular siguiente generación	209
7.1.33 PU33 El tipo de regla es semitotalista	210
7.1.34 PU34 Calcular densidad de población	210
7.1.35 PU35 Calcular entropía	211
7.1.36 PU36 Calcular polinomio característico	211
7.1.37 PU37 Calcular puntos de intersección	212
7.1.38 PU38 Graficar puntos de intersección	212
7.1.39 PU39 Puntos del tipo repulsivo	213
7.1.40 PU40 Puntos del tipo crítico	213
7.1.41 PU41 Puntos del tipo indiferente	214
7.1.42 PU42 Iniciar simulación	214
7.1.43 PU43 Detener simulación	214
7.1.44 PU44 Restablecer simulación	215
7.2 Pruebas integradoras	216
7.2.1 IT01 Configuración	217
7.2.2 IT02 Evolución	222
7.2.3 IT03 Resultado	225
IV Exploración y clasificación de reglas	229
8 Exploración de reglas	230
8.1 Regla 33938756	230
9 Reglas exploradas	233
9.1 Asociación de reglas y polinomios	234
9.2 Asociación de reglas y densidades	257
10 Clasificación	281
10.1 Clasificación de Wolfram	282

11 Casos de estudio	284
11.1 Analizando la regla 1688232692	284
11.2 Analizando la regla 2878016454	287
11.3 Analizando la regla 3367827048	290
11.4 Analizando la regla 3740243592	294
11.5 Analizando la regla 4133616598	297
11.6 Analizando la regla 3185432644	302
12 Conclusiones	308
12.1 Resultados	308
12.2 Limitaciones	309
12.3 Trabajo por realizar	309

Índice de figuras

Fig. 1.2.1	Divergencia exponencial a través del tiempo	16
Fig. 1.2.2	Tipos de atractores.	18
Fig. 2.3.1	Tipos de vecindades	24
Fig. 2.3.2	Evaluación semitotalista para la vecindad de Moore en un AC bidimensional.	25
Fig. 2.3.3	Evaluación semitotalista para la vecindad de von Neumann en un AC bidimensional.	25
Fig. 2.3.4	Evaluación totalista para la vecindad de Moore en un AC bidimensional.	26
Fig. 2.3.5	Evaluación totalista para la vecindad de von Neumann en un AC bidimensional.	26
Fig. 2.5.6	Generación de muestra [20].	29
Fig. 2.5.7	GDesign 2.0 de Umberto Roncoroni [21].	29
Fig. 2.5.8	Evolución de la regla 30 en un autómata unidimensional.	30
Fig. 2.8.9	Entropía para el caso de dos alternativas con probabilidades p y $(1-p)$	34
Fig. 2.9.10	Máquina de Turing simulada en Golly.	35
Fig. 2.9.11	Oscilador alcanzando la generación número 201,626,176 en menos de 1 minuto.	36
Fig. 2.9.12	Archivo de reglas en Golly	37
Fig. 2.9.13	Reglas en el simulador vNCASimulator	37
Fig. 2.9.14	Comparativa de software Golly y vNCASimulator	37
Fig. 2.9.15	ACOSXL21 para Mac OS X [32]	38
Fig. 2.9.16	ACOSXSTM para Mac OS X [33]	39
Fig. 2.9.17	ACOSXSTM para Windows NT ACOSXSTM [34]	39
Fig. 2.9.18	ACOSXV para OpenStep [35]	40
Fig. 2.9.19	ACOSXM para Mac OS X Server [36]	41
Fig. 2.9.20	Polinomio de campo promedio de Life por el software vNCASimulator	42
Fig. 2.9.21	Proyección en tres dimensiones	43
Fig. 2.9.22	Sitio web www.samcodes.co.uk	43
Fig. 2.9.23	Distintos patrones disponibles	44
Fig. 4.4.1	Diagrama de clases completo	51
Fig. 4.4.2	Diagrama de clases	52
Fig. 4.4.3	Diagrama de clases (continuación)	53
Fig. 4.4.4	Diagrama de calses (continuación)	54
Fig. 4.4.5	Macroproceso	56
Fig. 4.4.6	Subproceso - Draw	57
Fig. 4.4.7	Subproceso - Draw (continuación)	58

Fig. 4.4.8	Subproceso - Next generation	58
Fig. 4.4.9	Subproceso - Solicitar siguiente generación	59
Fig. 4.4.10	Subproceso - Calcular siguiente generación	60
Fig. 4.4.11	Subproceso - Calcular siguiente generación (continuación)	61
Fig. 4.4.12	Subproceso - Play/pause	61
Fig. 4.4.13	Subproceso - Establecer configuración inicial	62
Fig. 4.4.14	Subproceso - Establecer configuración inicial (continuación)	63
Fig. 4.4.15	Subproceso - Definir atributos del autómata celular	64
Fig. 4.4.16	Subproceso - Definir atributos del autómata celular (continuación)	65
Fig. 4.4.17	Subproceso - Crear autómata celular	66
Fig. 4.4.18	Subproceso - Configuración inicial del autómata celular	67
Fig. 4.4.19	Subproceso - Configuración inicial del autómata celular (continuación)	68
Fig. 4.4.20	Subproceso - Actualizar valores	69
Fig. 4.4.21	Subproceso - Simulación	70
Fig. 4.4.22	Subproceso - Definir tipo de regla	70
Fig. 4.4.23	Subproceso - Graficar polinomio característico	71
Fig. 4.4.24	Subproceso - Graficar polinomio característico (continuación)	71
Fig. 4.4.25	Subproceso - Clear	72
Fig. 4.4.26	Subproceso - Export	73
Fig. 4.4.27	Subproceso - Export (continuación)	74
Fig. 4.4.28	Subproceso - Import	75
Fig. 4.4.29	Subproceso - Import (continuación)	76
Fig. 5.1.1	PR01.1 Ventana de configuración - Acciones	116
Fig. 5.1.2	PR01.2 Ventana de configuración - Evaluación	117
Fig. 5.1.3	PR01.3 Espacio de evolución	118
Fig. 5.1.4	PR02.2 Ventana de configuración - Condiciones iniciales	119
Fig. 5.1.5	PR02.2 Espacio abierto	120
Fig. 5.1.6	PR02.3 Espacio cerrado	121
Fig. 5.1.7	PR03.1 Ventana de configuración - Condiciones iniciales	122
Fig. 5.1.8	PR03.2 Ventana de configuración - Espacio abierto	123
Fig. 5.1.9	PR04.1 Ventana de configuración - Acciones	125
Fig. 5.1.10	PR04.2 Ventana de configuración - Entropía	126
Fig. 5.1.11	PR04.3 Ventana de configuración - Población	127
Fig. 5.1.12	PR04.4 Ventana de configuración - Polinomio	128
Fig. 5.1.13	PR04.5 Proyección	129
Fig. 5.1.14	PR04.6 Ventana de configuración - Visualización del espacio	130
Fig. 6.1.1	IU-CU01 Definir la altura del espacio	132
Fig. 6.1.2	IU-CU01.1 Definir la anchura del espacio	133
Fig. 6.1.3	IU-CU02 Definir el tipo de vecindad como Moore	134
Fig. 6.1.4	IU-CU02.1 Definir el tipo de vecindad como von Neumann	135
Fig. 6.1.5	IU-CU03 Definir el tipo de regla como totalista	136
Fig. 6.1.6	IU-CU04 Cambiar el valor de la célula para una determinada configuración	137
Fig. 6.1.7	IU-CU04.1 Cambiar la densidad para una regla totalista aleatoria	138
Fig. 6.1.8	IU-CU04.1-1 Establecer regla aleatoria	139
Fig. 6.1.9	IU-CU04.2 Ingresar ID de la regla totalista	140
Fig. 6.1.10	IU-CU05 Restablacer valores de la regla totalista	141
Fig. 6.1.11	IU-CU06 Definir N_{min}	142
Fig. 6.1.12	IU-CU06.1 Definir N_{max}	143
Fig. 6.1.13	IU-CU06.2 Definir S_{min}	144
Fig. 6.1.14	IU-CU06.3 Definir S_{max}	145
Fig. 6.1.15	IU-CU07 Definir tipo de espacio como cerrado	146

Fig. 6.1.16	IU-CU07.1 Definir tipo de espacio como abierto	147
Fig. 6.1.17	IU-CU08 Establecer configuración inicial aleatoria	148
Fig. 6.1.18	IU-CU08.1 Establecer configuración inicial desde archivo	149
Fig. 6.1.19	IU-CU08.1-1 Importar archivo	150
Fig. 6.1.20	IU-CU08.1-2 Ventana de selección de archivo	151
Fig. 6.1.21	IU-CU09 Exportar objeto AC	152
Fig. 6.1.22	IU-CU09.1 Seleccionar directorio	153
Fig. 6.1.23	IU-CU10 Definir el tamaño de las células	154
Fig. 6.1.24	IU-CU11 Definir la velocidad de evolución del AC	155
Fig. 6.1.25	IU-CU12 Asignar valor del escalar	156
Fig. 6.1.26	IU-CU13 Estado del gradiente.	157
Fig. 6.1.27	IU-CU14 Estado de la rejilla	158
Fig. 6.1.28	IU-CU14-1 Espacio de evolución del AC	159
Fig. 6.1.29	IU-CU15 Definir el color de la rejilla	160
Fig. 6.1.30	IU-CU15.1 Etiqueta indicadora de la rejilla	161
Fig. 6.1.31	IU-CU15-1 Paleta de colores	162
Fig. 6.1.32	IU-CU15.2 Rejilla en el espacio de evolución del AC	163
Fig. 6.1.33	IU-CU16 Definir el color de las células vivas	164
Fig. 6.1.34	IU-CU16.1 Definir el color de las células muertas	165
Fig. 6.1.35	IU-CU16.2 Definir el color de las células de proyección	166
Fig. 6.1.36	IU-CU16.3 Etiqueta indicadora de colores de las células	167
Fig. 6.1.37	IU-CU16-1 Células vivas en el espacio de evolución del AC	168
Fig. 6.1.38	IU-CU16-2 Células muertas en el espacio de evolución del AC	169
Fig. 6.1.39	IU-CU16-3 Células de proyección en el espacio de evolución del AC	170
Fig. 6.1.40	IU-CU17 Cambiar una célula del espacio de evolución del AC	171
Fig. 6.1.41	IU-CU18 Ventana de configuración	172
Fig. 6.1.42	IU-CU19 Desplegar CA con la configuración inicial.	173
Fig. 6.1.43	IU-CU19.1 Espacio de evolución con la configuración inicial	174
Fig. 6.1.44	IU-CU21 Desplegar siguiente generación	175
Fig. 6.1.45	IU-CU21-1 Siguiente generación del AC	176
Fig. 6.1.46	IU-CU26 Graficar población	177
Fig. 6.1.47	IU-CU26.1 Gráfica de población	178
Fig. 6.1.48	IU-CU27 Graficar entropía	179
Fig. 6.1.49	IU-CU27.1 Gráfica de entropía	180
Fig. 6.1.50	IU-CU28 Graficar polinomio característico	181
Fig. 6.1.51	IU-CU28.1 Lista de puntos clasificados	182
Fig. 6.1.52	IU-CU28.2 Gráfica del polinomio característico	183
Fig. 6.1.53	IU-CU29 Graficar recta identidad	184
Fig. 6.1.54	IU-CU29.1 Gráfica del polinomio característico con recta identidad	185
Fig. 6.1.55	IU-CU30 Graficar puntos de intersección	186
Fig. 6.1.56	IU-CU30.1 Gráfica del polinomio característico con puntos de intersección del tipo atractivo	187
Fig. 6.1.57	IU-CU30.2 Gráfica del polinomio característico con puntos de intersección del tipo repulsivo	188
Fig. 6.1.58	IU-CU30.3 Gráfica del polinomio característico con puntos de intersección del tipo crítico	189
Fig. 6.1.59	IU-CU31 Iniciar simulación	190
Fig. 6.1.60	IU-CU32 Detener simulación	191
Fig. 6.1.61	IU-CU33 Restablecer simulación	192
Fig. 6.1.62	IU-CU33.1 Restablecer gráfica de población	193
Fig. 6.1.63	IU-CU33.2 Restablecer gráfica de entropía	194
Fig. 6.1.64	IU-CU33.3 Restablecer espacio de evolución del AC	195
Fig. 8.1.1	Gráfica del polinomio correspondiente a la regla 33938756	232
Fig. 11.1.1	Simulación de la regla 1688232692 con densidad inicial de 0.9%	284

Fig. 11.1.2 Simulación de la regla 1688232692 con densidad inicial de 10%	285
Fig. 11.1.3 Comparación de la regla y corteza de pino	286
Fig. 11.1.4 Polinomio característico de la regla 1688232692	286
Fig. 11.1.5 Población de la regla 1688232692 hasta la generación 1000	287
Fig. 11.1.6 Entropía de la regla 1688232692 hasta la generación 1000	287
Fig. 11.2.7 Polinomio característico de la regla 2878016454	288
Fig. 11.2.8 Simulación de la regla 2878016454 con una sola célula viva tras 500 generaciones	289
Fig. 11.2.9 Gráficas obtenidas con una sola célula viva tras 500 generaciones	289
Fig. 11.2.10 Simulación de la regla 2878016454 con una densidad inicial de 5% tras 500 generaciones	290
Fig. 11.2.11 Gráficas obtenidas densidad inicial del 10% tras 500 generaciones	290
Fig. 11.3.12 Polinomio característico de la regla 3367827048	291
Fig. 11.3.13 Simulación de la regla 3367827048 con una densidad inicial de 1% tras 100 generaciones	292
Fig. 11.3.14 Gráficas obtenidas densidad inicial del 1% tras 100 generaciones	292
Fig. 11.3.15 Simulación de la regla 3367827048 con una densidad inicial de 99% tras 200 generaciones	293
Fig. 11.3.16 Gráficas obtenidas densidad inicial del 99% tras 200 generaciones	293
Fig. 11.3.17 Simulación de la regla 3367827048 con una densidad inicial de 50% tras 1000 generaciones	294
Fig. 11.3.18 Gráficas obtenidas densidad inicial del 50% tras 1000 generaciones	294
Fig. 11.4.19 Polinomio de campo promedio	295
Fig. 11.4.20 Condición inicial al 5%, generación número 1000	296
Fig. 11.4.21 Gráficas obtenidas con una densidad inicial de 5% tras 1000 generaciones	296
Fig. 11.4.22 Condición inicial al 5%, generación número 100	297
Fig. 11.4.23 Gráficas obtenidas con una densidad inicial de 50% tras 100 generaciones	297
Fig. 11.5.24 Polinomio de campo promedio	298
Fig. 11.5.25 Condición inicial con una célula central, generación número 500	298
Fig. 11.5.26 Gráficas obtenidas con una densidad inicial de una célula tras 500 generaciones	299
Fig. 11.5.27 Condición inicial al 5%, generación número 50	300
Fig. 11.5.28 Gráficas obtenidas con una densidad inicial del 5% tras 50 generaciones	300
Fig. 11.5.29 Condición inicial al 90%, generación número 50	301
Fig. 11.5.30 Gráficas obtenidas con una densidad inicial del 90% tras 50 generaciones	301
Fig. 11.6.31 Polinomio de campo promedio	302
Fig. 11.6.32 Condición inicial al 0.1%, generación número 500	303
Fig. 11.6.33 Gráficas obtenidas con una densidad inicial del 0.1% tras 500 generaciones	303
Fig. 11.6.34 Condición inicial al 5%, generación número 500	304
Fig. 11.6.35 Gráficas obtenidas con una densidad inicial del 5% tras 500 generaciones	304
Fig. 11.6.36 Condición inicial al 80%, generación número 500	305
Fig. 11.6.37 Gráficas obtenidas con una densidad inicial del 80% tras 500 generaciones	305
Fig. 11.6.38 Glider con 2 configuraciones	306
Fig. 11.6.39 Glider con 2 configuraciones	306
Fig. 11.6.40 Glider con 3 configuraciones	306
Fig. 11.6.41 Colisión de gliders	307

Parte I

Introducción

La teoría de los autómatas celulares (en adelante, AC) surge con el libro *Theory of Self-reproducing automata* de John von Neumann a finales de la década de 1940.

Un AC es un modelo matemático para un sistema dinámico que evoluciona en pasos en tiempo discreto de acuerdo a una serie de reglas y configuraciones iniciales.

Este trabajo presenta un estudio sobre los AC en un espacio de dos dimensiones. Los AC bidimensionales están constituidos por un tablero (conocido como rejilla), donde cada una de las celdas representa una célula que puede tomar uno de los posibles estados definidos, dependiendo de la configuración inicial, las reglas que rigen al sistema, el estado actual en el que se encuentra y del estado que tengan las células (celdas) vecinas. Este tipo de autómatas trabajan bajo dos tipos de vecindad principalmente: la vecindad de Moore y la vecindad de von Neumann, las cuales tienen ocho y 4 vecinos respectivamente, sin contar la célula central.

Las ecuaciones que se deben seguir para aplicar una u otra vecindad así como sus características y repercusiones sobre el estado de las células se presentan a detalle en este documento.

Existe un AC bidimensional que ha dado paso a una infinidad de investigaciones por parte de matemáticos y divulgadores científicos: **El Juego de la vida**, mejor conocido por su nombre en inglés, *The Game Of Life*. El Juego de la Vida es, como ya se mencionó, un AC de dos dimensiones en el que se encuentran principalmente dos estados para las células: *vivo* o *muerto*. Las condiciones iniciales, así como las reglas con las que se haga la simulación sobre el mismo harán una pauta para realizar una exploración sobre las vecindades de Moore y de von Neumann, siendo estas últimas el objeto principal de estudio, así como su clasificación de acuerdo al comportamiento cualitativo que presenten: estable y homogéneo, estable periódico, pseudo-aleatorio o caótico, o complejo (clasificación del comportamiento cualitativo de los AC unidimensionales de Stephen Wolfram) utilizando la teoría de campo promedio para la obtención de polinomios característicos, que permitan agruparlos y asociarlos de acuerdo con las características de los mismos. Para ello se desarrollará un simulador de AC bidimensionales con proyección al espacio tridimensional; esta herramienta será de gran utilidad para estudiar el comportamiento de forma visual modificando las reglas y condiciones iniciales, también para obtener los polinomios característicos de una forma más rápida y hacer comparaciones más exactas a lo que se pudiera realizar sin ella. Dicho simulador tiene como objetivo principal trabajar bajo la vecindad de von Neumann en un espacio cerrado con reglas totalistas, sin embargo se hace una extensión para poder observar también el comportamiento con la vecindad de Moore, elegir si el tipo de espacio de evolución es abierto y trabajar con reglas semitotalistas o totalistas externas, mismas de las cuales se explica de forma general en qué consisten y cuáles son las diferencias.

Parte II

Estado del arte

1

Sistemas dinámicos

Un sistema dinámico es un sistema complejo capaz de representar modelos de la naturaleza que tienen una evolución en el tiempo, ya sea de forma discreta o continua. La evolución se lleva a cabo con base en estados; el sistema comienza en un estado x_0 y la evolución al estado siguiente depende del estado anterior; además, existe una ley que rige su evolución y si se conoce esta y el estado inicial es posible encontrar cualquier estado posterior [1].

Formalmente, un sistema dinámico es una función $\phi(t, x)$ definida para todo $t \in R$ y $x \in E \subset R^n$, donde R representa el espacio de los números reales y n cualquier dimensión para ese espacio, y describe cómo evolucionan los puntos $x \in E$ respecto al tiempo. Esta evolución debe seguir una ley descrita en forma de ecuación, donde el objetivo es hallar el valor de x en cualquier tiempo t de un dominio temporal determinado, es decir $X(t)$. Dependiendo del dominio temporal podemos trabajar en la dinámica discreta si el dominio también es discreto, en caso contrario, le corresponde el ámbito de la dinámica continua [2].

En Ec. 1.0.1 se puede observar que la ley que rige la evolución se encuentra dentro del dominio de los sistemas dinámicos discretos, esto se debe principalmente a que los valores que puede tomar t son discretos.

$$X(t+2) = X(t+1) + X(t), t \in \mathbb{N} \quad (\text{Ec. 1.0.1})$$

Observemos Ec. 1.0.2, se puede notar que la ley que describe la evolución se encuentra dentro del dominio de los sistemas dinámicos continuos, ya que t puede tomar cualquier valor real no negativo.

$$X''(t) + 2X'(t) + X(t) - t = 0, t \in [0, +\infty) \quad (\text{Ec. 1.0.2})$$

A continuación, se describe en qué consisten los modelos continuo y discreto.

1.1. Modelo continuo y discreto

Los sistemas dinámicos pueden clasificarse según el tiempo: en los que el tiempo varía de forma continua y en los que el tiempo transcurre discretamente.

1.1.1. Sistemas dinámicos a tiempo continuo

Para definir un sistema dinámico tenemos que partir de un campo vectorial en un espacio euclíadiano \mathbb{R}^n , o bien de una función que va de \mathbb{R}^n en sí mismo.

Sea X un campo vectorial definido en \mathbb{R}^n o en un abierto de \mathbb{R}^n , C^∞ . A este campo vectorial $X : \mathbb{R}^n \rightarrow \mathbb{R}^n$ se le puede asociar un sistema de ecuaciones diferenciales ordinarias que a su vez puede ser expresado de forma vectorial como Ec. 1.1.1:

$$\dot{x} = X(x) \quad (\text{Ec. 1.1.1})$$

donde $x \in \mathbb{R}^n$ y el punto denota la derivada con respecto al tiempo [3]. Una condición $x_0 \in \mathbb{R}^n$ determina una solución, es decir una curva diferenciable $x = \phi(t)$ en \mathbb{R}^n tal que $\phi(0) = 0$ y su derivada satisface Ec. 1.1.2

$$\frac{d\phi}{dt}(t) = X(\phi(t)) \quad (\text{Ec. 1.1.2})$$

Esta curva se encuentra definida en un intervalo maximal $I \subset \mathbb{R}$ que contiene al 0.

Este campo vectorial define un sistema dinámico a tiempo continuo cuando el objeto de interés es el estudio global cualitativo de sus soluciones.

1.1.2. Sistemas dinámicos a tiempo discreto

A diferencia de los sistemas dinámicos a tiempo continuo en vez de tener una curva ahora se tiene un conjunto discreto de puntos que puede ser finito, es decir que las soluciones de una ecuación diferencial dada no tienen por qué estar definidas para todo $t \in \mathbb{R}$. Sea f una función real de una variable real, la ecuación en diferencias o sistema dinámico a tiempo discreto correspondiente es

$$x_{k+1} = f(x_k) \quad (\text{Ec. 1.1.3})$$

donde $k = 0, 1, 2, 3, \dots$

Dada una condición inicial $x_0 \in \mathbb{R}$, una solución consistirá generalmente en una sucesión de puntos $x_0, x_1, x_2, x_3, \dots$ que puede construirse iterando a partir de la Ec. 1.1.3. En general, si F es una función continua de \mathbb{R}^n en sí mismo, o con dominio abierto de \mathbb{R}^n , podemos definir el sistema dinámico a tiempo discreto como Ec. 1.1.4

$$x_{k+1} = F(x_k), k = 0, 1, 2, 3, \dots \quad (\text{Ec. 1.1.4})$$

Una solución de este sistema queda determinada al fijar una condición inicial x_0 en el dominio F . Se puede observar que, para funciones de una variable, la función siempre será una sucesión de puntos si el dominio de F es todo \mathbb{R}^n [3]. Si denotamos por $F^{(k)}$ a la composición de F consigo mismo k veces, podemos escribir explícitamente cualquier elemento en términos de x_0 como se muestra en Ec. 1.1.5

$$x_k = F^{(k)}(x_0), k = 0, 1, 2, 3, \dots \quad (\text{Ec. 1.1.5})$$

En el presente trabajo se estará utilizando el dominio de los sistemas dinámicos discretos, ya que nos permite describir el comportamiento de los AC; esto se debe a que los estados de las células son alterados de un instante a otro en unidades de tiempo discreto. Este es un tema que requiere ser analizado a profundidad, por lo que será abordado en el próximo capítulo referente a los AC.

1.2. Teoría del caos

El caos es un tipo de comportamiento aperiódico a largo plazo en un sistema determinístico, acotado en el espacio de estado, irregular y presenta una alta sensibilidad a las condiciones iniciales.

Existen diferentes características que ayudan a describir este comportamiento y se detallan a continuación.

1.2.1. Determinismo

Un sistema dinámico es un modelo matemático determinista, en donde el tiempo (como se ha estudiado anteriormente) puede ser una variable continua o discreta. Se considera que para que un modelo matemático sea determinista debe presentar una evolución única, es decir que un estado dado de un modelo sigue siempre la misma historia de transiciones de estado[4].

1.2.2. Acotado

Otra característica del comportamiento caótico es que está acotado, esto quiere decir que el sistema no se dispara hasta el infinito. En el modelo discreto, siempre que la condición inicial esté dentro del intervalo $(0, 1)$, el resultado siempre estará en ese intervalo [4]; el estado nunca tomará valores más altos o más bajos. La caracterización de los límites para el dominio del modelo se conocen como *condiciones límite o de acotamiento*.

1.2.3. Irregularidad

La irregularidad, también conocida como comportamiento aperiódico significa que las variables del sistema nunca repiten ningún valor de manera regular. El comportamiento caótico, comienza de manera irregular y se mantiene irregular a lo largo del tiempo.

Cabe mencionar que a pesar de que el caos se comporta de forma aperiódica, no es aleatorio, es determinístico.

1.2.4. Sensible a las condiciones iniciales

Una de las características más importantes del caos y también más estudiadas es la dependencia sensitiva a condiciones iniciales (popularmente conocido como el “efecto mariposa” de Lorenz)[7], se refiere a que si se realiza un pequeño cambio en una variable tendrá un gran impacto impredecible en la evolución del sistema en cuestión. Matemáticamente, estos cambios siguen una dinámica no lineal. Dicho de otra forma, dos trayectorias que parecen muy similares eventualmente divergirán hasta un punto en el que ambas trayectorias sean totalmente diferentes.

Sean N_0 y M_0 dos condiciones iniciales diferentes, se define $d(M_0, N_0)$ como la distancia entre M_0 y N_0 . Dado que M_0 y N_0 son puntos en el espacio tridimensional (X, Y, Z), la distancia entre ellas es la distancia Euclídea Ec. 1.2.1

$$d(M, N) = \sqrt{(X_M - X_N)^2 + (Y_M - Y_N)^2 + (Z_M - Z_N)^2} \quad (\text{Ec. 1.2.1})$$

Después de un tiempo t , ambos puntos habrán evolucionado a M_t y N_t respectivamente. La dependencia sensitiva afirma que $d(M_t, N_t)$ crece exponencialmente a través del tiempo para algún λ . La velocidad a la que se produce dicha divergencia se mide con exponentes de Lyapunov (λ) [4], tal que (Ec. 1.2.2)

$$d(M_t, N_t) = e^{\lambda \cdot t} \cdot d(M_0, N_0) \quad (\text{Ec. 1.2.2})$$

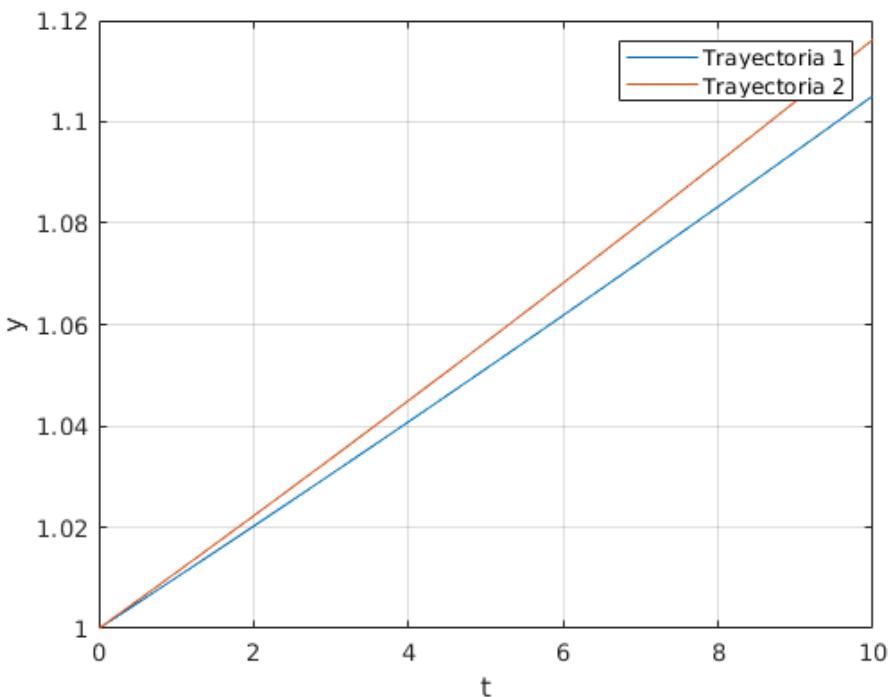


Fig. 1.2.1: Divergencia exponencial a través del tiempo

Fuente: “Modeling Life. The Mathematics of Biological Systems” [4].

En la Fig. 1.2.1, se puede apreciar que para un tiempo $t = 0$ parece que ambas trayectorias se encuentran en un mismo punto, pero hay un pequeño cambio aplicado entre las trayectorias, para la primera se utilizó $\lambda = 0.01$ mientras que en la segunda trayectoria se utilizó $\lambda = 0.011$, es decir un cambio de 0.001 y al paso del tiempo (t) la distancia entre ambas trayectorias aumenta de tal forma que se puede observar que son completamente diferentes.

El número de exponentes de Lyapunov depende de las dimensiones del espacio-fase. Para que un sistema sea considerado caótico al menos uno de los exponentes debe ser positivo (esto hace que exista un horizonte de predictibilidad finito), y la suma de todos ellos debe ser negativa, para que exista un atractor estable. Por otra parte, si la suma de los exponentes es positiva el sistema tiende al azar, y si todos los exponentes son cero o negativos los sistemas son lineales, es decir, predictibles [5].

1.2.5. Imprevisibilidad

Esta característica está dada precisamente por la dependencia sensitiva a las condiciones iniciales, ya que, al realizar un cambio, así sea infinitesimal, el comportamiento del sistema será completamente distinto. Es decir, que si se simula el sistema bajo las mismas condiciones a excepción de que para una primera simulación se utiliza la variable $X_0 = 0.01$, y para una segunda $X_0 = 0.011$, esa milésima provocará que las trayectorias tengan comportamientos totalmente diferentes. Para observar lo impredecible que puede ser un sistema tras aplicar un cambio infinitesimal se puede analizar la gráfica anterior (Fig. 1.2.1) donde el cambio realizado provoca que las trayectorias tomen distintos puntos a través del tiempo.

1.2.6. Atractores caóticos

Un atractor A de un sistema dinámico se define según Ec. 1.2.3

$$V : X \rightarrow T(X) \quad (\text{Ec. 1.2.3})$$

como un subconjunto A de X , el cual tiene la propiedad que, para un conjunto grande de condiciones iniciales, cada trayectoria tiende a A como $T \rightarrow \infty$.

Existen cuatro tipos de atractores los cuales son mostrados en Fig. 1.2.2 y a continuación se detalla el significado de cada uno de ellos:

- a Atractores de punto, también conocidos como puntos de equilibrio estables, representan un comportamiento que es estático o se acerca a él.
- b Atractores de ciclo límite, u órbitas cerradas estables, representan el comportamiento que es periódico o se acerca a él.
- c Atractores toro-límite, puede haber más de una frecuencia en la trayectoria periódica del sistema a través de un ciclo límite. Si dos de esas frecuencias forman una relación irracional, la trayectoria ya no se cierra y el ciclo límite pasa a ser un toro-límite.
- d Atractores extraños, o caóticos, representan un tercer tipo de movimiento, aparte del comportamiento de equilibrio y el comportamiento oscilatorio [4, 6, 8].

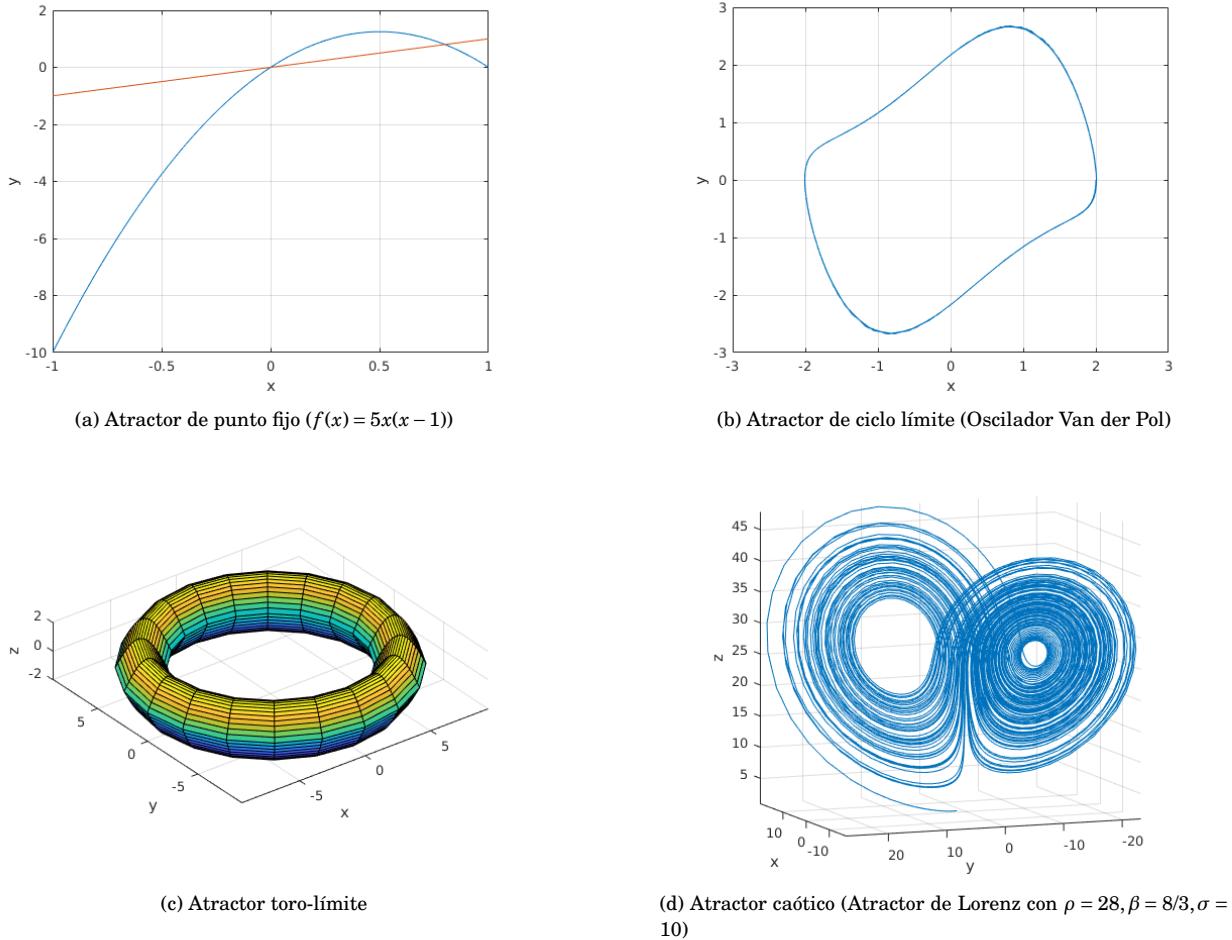


Fig. 1.2.2: Tipos de atractores.

Fuente: Realización propia.

Un *atractor extraño*, nombrado por el físico matemático David Ruelle y el matemático Floris Takens en 1971, es aquel que tiene un movimiento aperiódico y es muy sensible a las condiciones iniciales. Los atractores extraños son complejos, con forma tridimensional y tienen una estructura detallada en todos los niveles de magnificación; es decir, al aumentar cualquier región pequeña del atractor se podrá encontrar que la porción ampliada se ve idéntica a la región de tamaño regular [4, 8]. La repetición continua de este proceso producirá los mismos resultados. Esta estructura auto-similar se repite en escalas arbitrarias pequeñas.

Matemáticamente, este tipo de atractores son representados por medio de fractales. En muchas ocasiones surgen cuando diferentes ciclos límite y puntos fijos tipo silla se encuentran en el sistema. Bajo ciertas condiciones los repulsores enviarán la trayectoria a infinito; sin embargo, habrá situaciones en las que la trayectoria se mantenga viajando de repulsor a repulsor, sin tender a infinito, haciendo una órbita aperiódica. En síntesis, un sistema con comportamiento caótico evoluciona dado un atractor extraño y por lo tanto el sistema presenta un comportamiento aperiódico a largo plazo.

La teoría del caos es el intento del conocimiento de lo que va a suceder, no de lo que es o de lo que permanecerá. Con ella lo desconocido se interpreta desde un punto de vista global, que se ve afectado en el tiempo. Hablar de caos es hablar de procesos irreversibles, dado que su comportamiento es impredecible y puede presentar cambios exponenciales de acuerdo al tiempo [9].

1.3. Sistemas complejos

“La palabra *complejo* tiene su origen en la raíz latina *plexus* que se puede traducir como *tejer, enredar* o incluso *torcer*. De ahí se deriva la noción latina *plexus* que significa *entrelazado*. Si se considera que el prefijo latino *con* significa *junto* o *todo*, se tiene que la palabra *complejo* implica un *todo entrelazado o lo que está entrelazado junto*”[10].

Para entender el comportamiento de un sistema complejo se debe entender que no solo es importante estudiar el comportamiento de cada una de las partes que componen al sistema, también es necesario conocer cómo actúan en conjunto para poder ver el comportamiento del todo.

1.3.1. Complejidad

La complejidad dependerá del nivel de detalle que se requiera para describir un sistema. Si se tiene un sistema que puede tomar muchos estados posibles, pero se quiere especificar en qué estado se encuentra realmente, entonces el número de dígitos binarios (bits) que se necesitan para especificar dicho estado está relacionado con el número de estados posibles. Sea Ω el número de estados, entonces el número de bits que se requieren para almacenar la información es Ec. 1.3.1

$$I = \log_2(\Omega) \quad (\text{Ec. 1.3.1})$$

Para representar cada estado de forma única se requerirán tantos números como estados tenga el sistema; por lo tanto, el número de estados de la representación debe ser el mismo que el número de estados del sistema. Para una cadena de N bits existen 2^N estados posibles, formalmente se describe como Ec. 1.3.2

$$\Omega = 2^N \quad (\text{Ec. 1.3.2})$$

lo que implica que N es igual a I (Ec. 1.3.1) [11].

A continuación, se presentan las propiedades asociadas con un sistema complejo.

1.3.2. No linealidad

La no linealidad es considerada esencial para un sistema complejo. Un sistema lineal se modela por medio de ecuaciones que dependen proporcionalmente de la variable de estado del sistema y , y, posiblemente de sus derivadas. Únicamente se admite una función lineal que satisface la ley de proporcionalidad. Si el sistema no cumple con esta característica, se dice que el sistema es **no lineal** [10, 12].

Se ha observado que esta característica está presente por ejemplo en el caos, para las ecuaciones de movimiento puede haber grandes consecuencias si hay pequeñas variaciones en los valores de las condiciones iniciales.

1.3.3. Retroalimentación

La retroalimentación es una condición importante para los sistemas complejos. Se dice que una parte de un sistema recibe retroalimentación cuando la forma en que las otras partes interactúan con ésta en un estado posterior depende de cómo interactuó con ellas en un estado anterior [12].

Comúnmente se utiliza la retroalimentación para realizar corrección de errores.

1.3.4. Orden espontáneo

Una idea fundamental en la investigación de sistemas complejos es la de ordenar el comportamiento de un sistema que surge del conjunto de una gran cantidad de interacciones no coordinadas entre sus partes. El orden espontáneo es la aparición natural de orden del aparente caos a través de la autoorganización[12], es decir, no requiere de la intervención de un ente externo para llegar al orden.

1.3.5. Robustez y falta de control central

Se dice que el orden es robusto en los sistemas complejos porque al estar distribuido y no ser producido de forma central, es estable bajo perturbaciones del sistema. Por otra parte, un sistema controlado centralmente es vulnerable al mal funcionamiento de algunos componentes clave.

A pesar de que la falta de control central es una característica de los sistemas complejos, no es suficiente para describir la complejidad porque también puede estar presente en los sistemas no complejos.

1.3.6. Emergencia

La emergencia se refiere al surgimiento de estructuras, patrones y propiedades nuevas y coherentes durante el proceso de autoorganización en los sistemas complejos (Goldstein, 1999) [14]. Cuando un sistema tiene una propiedad emergente global, el comportamiento de una parte pequeña es diferente de forma aislada que cuando está interactuando (forma parte) con el sistema. En general, se asume que la emergencia es un fenómeno sorprendente a nivel del comportamiento global del sistema.

Un ejemplo de esta característica es que nuestro cerebro está compuesto por neuronas. Las neuronas no tienen mente, pero nosotros sí. La mente emerge de las interacciones entre nuestras neuronas, nuestro cuerpo y nuestro entorno. Pero, si una neurona se activa, no se puede saber a qué pensamiento pertenece esa activación. En el Juego de la Vida de Conway: las estructuras que se mueven en el espacio no están especificadas en las reglas del juego: son un producto emergente de las interacciones locales entre las celdas. Si se observa una celda por separado, no se puede saber si es parte de una estructura de mayor escala o no[13].

El estudio del comportamiento de *El Juego de la Vida* es de gran importancia para este trabajo, por lo cual será abordado con más detalle en el siguiente capítulo.

1.3.7. Organización jerárquica

El resultado final de todas las características mencionadas arriba para los sistemas complejos es una entidad que se encuentra organizada en una variedad de niveles de estructura y propiedades que interactúan con la parte superior e inferior y exhiben varios tipos de simetría, orden y comportamiento periódico.

Un ejemplo de esto es el cosmos con su compleja estructura de átomos, moléculas, gases, líquidos,..., y finalmente estrellas y galaxias, cúmulos y supercúmulos. Toda esta estructura obedece a una jerarquía en la que cada una de sus partes interactúa con un nivel arriba y uno por debajo de ella [12].

En conclusión, muchos elementos individuales tienen un comportamiento cuando son estudiados de forma separada, este comportamiento puede o no mantenerse al ser estudiados como partes que interactúan en un todo, lo que permite generar sistemas complejos.

2

Autómatas Celulares

2.1. Antecedentes históricos

Los AC son construcciones muy simples que aparecieron en la década de los 50. La forma en la que se dieron a conocer fue gracias a John von Neumann mientras trataba de desarrollar un modelo abstracto de autorreproducción biológica, el cual era un gran tema de discusión en esa época. Primeramente, von Neumann comenzó a pensar en modelos en tres dimensiones que eran descritos con ecuaciones diferenciales parciales. Posteriormente cambió de parecer y pensó en robótica e imaginó que era posible implementar algún ejemplo utilizando un juego de construcción. Análogamente haciendo uso de circuitos electrónicos él pudo notar que utilizar un modelo en dos dimensiones debía ser suficiente.

En 1951 por sugerencia de Stanislaw Ulam, von Neumann simplificó su modelo y finalmente terminó con la creación de un AC en dos dimensiones. El AC fue construido en 1952 y tenía 29 colores por célula y reglas bastante complicadas, las cuales permitían emular las operaciones que realizaban los componentes de una computadora y varios dispositivos mecánicos. Para probar matemáticamente la posibilidad de la autorreproducción, von Neumann describe la construcción de una configuración de 200,000 células que debían reproducirse a sí mismas.

A causa del arduo trabajo que realizó von Neumann surgieron dos temas a discutir bastante importantes: el primero surge en los años 60 y fue el incremento de la discusión sobre la construcción de autómatas autorreproducibles, y el segundo fue un intento de capturar aún más la esencia de la autorreproducción mediante estudios matemáticos de las propiedades detalladas de un AC.

Mientras transcurría el tiempo en los 60s se descubrieron construcciones para AC más simples que tienen la capacidad de autorreproducción y son computacionalmente universales.

A finales de los 50s los AC también eran vistos como computadoras en paralelo y particularmente en los 60s surgieron bastantes teoremas técnicos bastante detallados que probaron formalmente las capacidades computacionales de estos. Al término de los 60s se realizaron muchos intentos por encontrar una relación entre AC y sistemas dinámicos, pero la relación entre ambos no sería una realidad sino hasta una década después. Para mediados de los 70s el trabajo sobre AC se volvería complicado y el interés en ellos sufrió una disminución sobre todo en países como Rusia y Japón. En los años 70s gracias a John Horton Conway surge el AC más famoso conocido hasta ahora, *The game of life*. Este AC tiene un espacio de evoluciones bidimensional, además cada célula posee dos posibles estados y reglas bastante simples. Este AC es computacionalmente universal lo que quiere decir que es posible simular una máquina de Turing. [15].

2.2. Conceptos básicos

Los AC son idealizaciones matemáticas de sistemas naturales. Un AC consiste en una red discreta de células, cada una de estas células toma un valor entero finito. Los valores de las células se obtienen en pasos

discretos de acuerdo a reglas deterministas que especifican el valor de la célula en términos de los valores que poseen sus células vecinas [16].

2.3. AC en 1D, 2D y 3D

2.3.1. AC en una dimensión

Sean \mathbb{Z} el conjunto de enteros y \mathbb{Z}^+ el conjunto de enteros positivos, entonces el espacio de evoluciones en una dimensión se representa como una sucesión de elementos que determinarán un arreglo unidimensional; sea c_i la i -ésima posición en el arreglo donde $i \in \mathbb{Z}$, cada una de las posiciones dentro del arreglo es llamada célula y estas células toman elementos del conjunto Σ , además $\Sigma \in \mathbb{Z}^+$. Este conjunto representa la cantidad de estados que puede manejar el autómata, por lo tanto, podemos afirmar que $c_i \in \Sigma$, es decir, cada célula toma un elemento de Σ . Se conoce como *configuración* a un conjunto de células. Wolfram representa a los autómatas de una dimensión con dos parámetros k y r donde k es el número de estados de Σ y r es el número de vecinos que posee una célula central (c_i). Los *vecinos* son las células ubicadas a la derecha e izquierda de la célula central c_i . El conjunto de vecinos a la izquierda y derecha de c_i es denominado *vecindad*.

Siendo así se puede representar a estos AC como una cuádrupla:

$$\{\Sigma, r, \phi, C_i\}$$

donde:

- Σ es el conjunto de estados
- r es el número de vecinos respecto a la célula central
- ϕ es la función de transición
- C_i la configuración inicial del sistema

Es posible conocer el número de estados Σ si se calcula su cardinalidad $|\Sigma| = k$, por tanto $k \in \mathbb{Z}^+$ y además Σ es numerable, es decir, $\Sigma = \{0, 1, \dots, k - 1\}$

Clases de Wolfram

Stephen Wolfram ha realizado una ardua investigación sobre los AC; uno de sus trabajos más destacados es sobre los AC unidimensionales, donde no solo estudió el comportamiento de las 256 reglas, sino que introduce a toda una clasificación de éstas de acuerdo a su comportamiento y complejidad. En el libro “A new kind of science” [18] Wolfram describe cuatro tipos de clases que se listan a continuación:

Clase I. Evolución a una configuración estable y homogénea: El comportamiento es muy simple, casi todas las condiciones iniciales conducen exactamente al mismo estado final. Se considera que la evolución es de orden o quietud.

Clase II. Evolución a segmentos periódicos constantes: Existen diferentes estados finales, todos ellos consisten en un conjunto de estructuras simples que se mantienen siempre igual o que se repiten cada cierto número de evoluciones.

Clase III. Evolución caótica: El comportamiento es un poco más complejo, existe cierta aleatoriedad y siempre en algún nivel se logran ver estructuras como triángulos.

Clase IV. Evolución a segmentos caóticos aislados que presentan un comportamiento complejo: Es una combinación entre las clases I, II y III, es decir, hay presencia de orden y aleatoriedad, se pueden encontrar patrones que por sí solos son muy simples, pero si se observa su interacción se aprecia que lo hace de una forma muy compleja.

Cada una de estas clases tiene su analogía con los sistemas dinámicos cuyo comportamiento está descrito en la teoría del caos mediante los atractores. De esta forma, la clase I que tiende a estabilizarse en un punto se puede comparar con los sistemas dinámicos que evolucionan en un punto fijo, para la clase II hay una asociación con los que evolucionan en ciclo límite que contienen principalmente configuraciones periódicas, en la clase III se le asocia con los que corresponden a los ciclos límite aperiódicos, es decir con los caóticos o atractores extraños; mientras que los clase IV al ser una combinación de las clases I, II y III se puede comparar con ambos aunque de acuerdo con Wolfram la clasificación para ésta es indecidible [17, 18].

2.3.2. AC en dos dimensiones

Los AC en dos dimensiones tienen como espacio de evolución un plano cartesiano. En este tipo de autómatas la función de transición tiene dos tipos de vecindades: vecindad de Moore y vecindad de von Neumann, las cuales se muestran en Fig. 2.3.1.

2.3.3. Vecindades

Formalmente un AC está definido por el producto cartesiano $\mathbb{Z} \otimes \mathbb{Z}$ siendo así, una célula $c_{i,j} \in \mathbb{Z} \otimes \mathbb{Z}$ está conectada a una célula $\{(c_{i+k_1}, c_{j+k_2}) : \text{Max}|k_1|, |k_2| \leq 1\}$, es decir, la vecindad de Moore donde $i, j, k \in \mathbb{Z}$. La vecindad de Moore se forma por una célula central $(c_{i,j})$ y ocho vecinos alrededor de dicha célula. Sea $\Sigma = \{0, 1\}$ el conjunto de estados, \mathcal{V} es la vecindad isotrópica, por tanto, el valor de \mathcal{V} para la vecindad de Moore es $\mathcal{V} = 8$ y sea c_0 la célula central donde $c_0 = c_{i,j}$ y las células $c_1, \dots, c_V = c_{i-1,j-1}, \dots, c_{i+1,j+1}$ son sus vecinos para toda $c_i \in \Sigma$. La función ϕ define la transformación local, las variables N_{min} y S_{min} indican el número mínimo de células ocupadas por el estado 1 en \mathcal{V} y las variables N_{max} y S_{max} el número máximo de células ocupadas por el estado 1 en \mathcal{V} en un tiempo t . Siendo así, si $c_0 = 1$ en un tiempo t entonces $c_0 = 1$ en el tiempo $t + 1$ si $N_{min} \leq \sum_{i=1}^V c_i \leq N_{max}$; por otra parte si $c_0 = 1$ en un tiempo t entonces $c_0 = 1$ en un tiempo $t + 1$ si $S_{min} \leq \sum_{i=1}^V c_i \leq S_{max}$ formalmente Ec. 2.3.1 [19].

$$\phi(c_0, c_1, \dots, c_V) = \begin{cases} 1 & \text{si } \begin{cases} c_0 = 0 & \text{y } N_{min} \leq \sum_{i=1}^V c_i \leq N_{max} \\ c_0 = 1 & \text{y } S_{min} \leq \sum_{i=1}^V c_i \leq S_{max} \end{cases} \\ 0 & \text{otro caso} \end{cases} \quad (\text{Ec. 2.3.1})$$

Para el caso de la vecindad de von Neumann se tiene algo análogo a lo que se encuentra en la ecuación (Ec. 2.3.1), pero en el caso de esta vecindad las células diagonales no son consideradas, solo se toman en cuenta las células ortogonales a la célula central, por tanto, la función de transición ϕ tiene cuatro vecinos y se puede decir que $\mathcal{V} = 4$. Sea $\Sigma = \{0, 1\}$ el conjunto de estados, la célula central $c_0 = c_{i,j}$ y $c_1 = c_{i-1,j}, c_2 = c_{i,j-1}, c_3 = c_{i+1,j}, c_4 = c_{i,j+1}$ son los vecinos o la vecindad isotrópica para toda $c_i \in \Sigma$. En la ecuación (Ec. 2.3.2) ϕ define la transformación local; las variables N_{min} y S_{min} indican la cantidad mínima de células ocupadas por el estado 1 en \mathcal{V} y las variables N_{max} y S_{max} el número máximo de células ocupadas por el estado 1 en \mathcal{V} en un tiempo t . Además, si $c_0 = 0$ en el tiempo t , entonces $c_0 = 1$ en el tiempo $t + 1$ si $N_{min} \leq \sum_{i=1}^V c_i \leq N_{max}$ y también si $c_0 = 1$ en un tiempo t entonces $c_1 = 1$ en un tiempo $t + 1$ si $S_{min} \leq \sum_{i=1}^V c_i \leq S_{max}$ formalmente Ec. 2.3.2[19].

$$\phi(c_0, \dots, c_V) = \begin{cases} 1 & \text{si } \begin{cases} c_0 = 0 & \text{y } N_{min} \leq \sum_{i=1}^V c_i \leq N_{max} \\ c_0 = 1 & \text{y } S_{min} \leq \sum_{i=1}^V c_i \leq S_{max} \end{cases} \\ 0 & \text{otro caso} \end{cases} \quad (\text{Ec. 2.3.2})$$

En Fig. 2.3.1 podemos observar ambas vecindades.

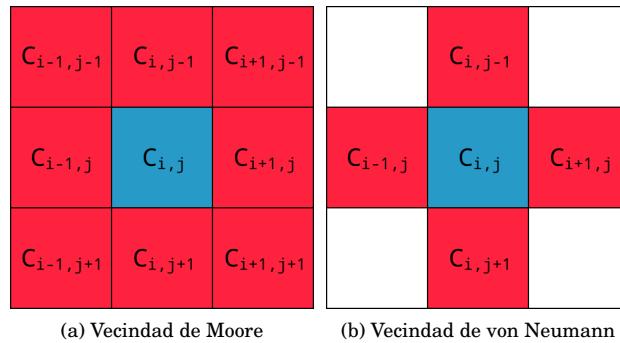


Fig. 2.3.1: Tipos de vecindades

Fuente: Realización propia.

2.3.4. Tipos de reglas

Existen diferentes tipos de reglas que se pueden aplicar a los AC.

A continuación se muestra una tabla comparativa del número de reglas que existen para vecindades del tipo von Neumann y Moore en AC con dos estados por célula.

Tipo de regla	Tipo de vecindad	
	von Neumann (5 vecinos)	Moore (9 vecinos)
General	$2^{32} \approx 4 \times 10^9$	$2^{512} \approx 10^{154}$
Totalista	$2^5 = 32$	$2^9 = 512$
Totalista externa	$2^{10} = 1024$	$2^{18} \approx 3 \times 10^5$

Tabla 2.1: Tabla comparativa del número total de reglas para células con dos estados [26].

Las siguientes definiciones se pueden generalizar, pero en este caso se comprende solamente para AC de dos dimensiones con dos estados por célula. Cabe mencionar que en las siguientes imágenes el color rojo indica que esas células forman parte de la evaluación para la regla en cuestión.

Reglas semitotalistas

Una regla semitotalista, también conocidas como totalista-externa, considera los valores de las células que se encuentran en el vecindario, es decir, alrededor de la célula que se está evaluando para actualizar el valor de la célula central denotada como $c_{i,j}$.

Regla semitotalista en la vecindad de Moore

$C_{i-1,j-1}$	$C_{i,j-1}$	$C_{i+1,j-1}$
$C_{i-1,j}$	$C_{i,j}$	$C_{i+1,j}$
$C_{i-1,j+1}$	$C_{i,j+1}$	$C_{i+1,j+1}$

Fig. 2.3.2: Evaluación semitotalista para la vecindad de Moore en un AC bidimensional.

Fuente: Realización propia.

Para una evaluación del tipo Moore, se sabe que hay 8 células vecinas, por lo cual el valor actualizado de la célula $c_{i,j}$ podemos observar esto en la Fig. 2.3.2 y lo anterior queda denotado formalmente como la Ec. 2.3.3

$$c_{i,j} = c_{i-1,j-1} + c_{i-1,j} + c_{i-1,j+1} + c_{i,j-1} + c_{i,j+1} + c_{i+1,j-1} + c_{i+1,j} + c_{i+1,j+1} \quad (\text{Ec. 2.3.3})$$

Regla semitotalista en la vecindad de von Neumann

	$C_{i,j-1}$	
$C_{i-1,j}$	$C_{i,j}$	$C_{i+1,j}$
	$C_{i,j+1}$	

Fig. 2.3.3: Evaluación semitotalista para la vecindad de von Neumann en un AC bidimensional.

Fuente: Realización propia a partir de la definición de la vecindad.

Al aplicar la regla semitotalista en la vecindad de von Neumann, sabemos que solo se consideran 4 células que son las que se encuentran alrededor de $c_{i,j}$ podemos observar esto en la Fig. 2.3.3 y lo podemos denotar como la Ec. 2.3.4

$$c_{i,j} = c_{i-1,j} + c_{i-1,j-1} + c_{i,j+1} + c_{i+1,j} \quad (\text{Ec. 2.3.4})$$

Por ello, para una vecindad del tipo von Neumann se pueden encontrar 32 configuraciones diferentes, ya que se evalúan todas las posibilidades para dos estados[26].

Regla totalista

La reglas totalistas además de considerar los valores de las células que se encuentran en el vecindario de la célula que se está evaluando, toma en cuenta el valor que tenía previamente la célula central para actualizar el valor de la siguiente generación [26].

$C_{i-1,j-1}$	$C_{i,j-1}$	$C_{i+1,j-1}$
$C_{i-1,j}$	$C_{i,j}$	$C_{i+1,j}$
$C_{i-1,j+1}$	$C_{i,j+1}$	$C_{i+1,j+1}$

Fig. 2.3.4: Evaluación totalista para la vecindad de Moore en un AC bidimensional.

Fuente: Realización propia a partir de la definición de la vecindad.

Regla totalista en la vecindad de Moore

De la definición anterior se puede observar que para realizar la evaluación solamente es necesario agregar una célula para realizar la suma y actualizar al nuevo valor de la célula $c_{i,j}$ como se muestra en Fig. 2.3.4 y con lo cual se obtiene Ec. 2.3.5

$$c_{i,j} = c_{i-1,j-1} + c_{i-1,j} + c_{i-1,j+1} + c_{i,j-1} + c_{i,j+1} + c_{i+1,j-1} + c_{i+1,j} + c_{i+1,j+1} + c_{i,j} \quad (\text{Ec. 2.3.5})$$

Regla totalista en la vecindad de von Neumann

	$C_{i,j-1}$	
$C_{i-1,j}$	$C_{i,j}$	$C_{i+1,j}$
	$C_{i,j+1}$	

Fig. 2.3.5: Evaluación totalista para la vecindad de von Neumann en un AC bidimensional.

Fuente: Realización propia a partir de la definición de la vecindad.

De la misma forma, esto aplica para la vecindad de von Neumann, por lo que se agrega el valor que tenía la célula $c_{i,j}$ tal y como se muestra en Fig. 2.3.5 con lo cual se obtiene la Ec. 2.3.6

$$c_{i,j} = c_{i-1,j} + c_{i-1,j-1} + c_{i,j+1} + c_{i+1,j} + c_{i,j} \quad (\text{Ec. 2.3.6})$$

Como se ha mencionado anteriormente, debido a que el comportamiento de reglas bajo la vecindad del tipo von Neumann no ha sido ampliamente explorado, se hará el análisis en este tipo de vecindad con reglas totalistas y totalistas externas pretendiendo obtener una clasificación por medio de la aplicación de la teoría del campo promedio, su comportamiento y la entropía.

2.3.5. AC en tres dimensiones

Los AC en tres dimensiones han sido ampliamente analizados por Bays. Su estudio se enfoca principalmente en encontrar una regla de evolución en tres dimensiones que sea la sucesora de *Life* en el espacio tridimensional.

Si bien los AC en dos dimensiones son difíciles de representar, en tres dimensiones el problema crece exponencialmente, es por ello que existe muy poco trabajo sobre el estudio de este espacio.

Sea $\Sigma = \{0, 1\}$ el conjunto de estados, el espacio de evoluciones en tres dimensiones se determina por el producto $\mathbb{Z} \otimes \mathbb{Z} \otimes \mathbb{Z}$. Al igual que los AC en dos dimensiones, en autómatas de tres dimensiones también se utilizan reglas de evolución semitotalísticas y la función de transición ϕ solo utiliza la vecindad de Moore. Para este caso se puede observar que existe una célula central y 26 vecinos alrededor de ésta por lo que $\mathcal{V} = 26$, por lo tanto, una vecindad en tres dimensiones está formada por 27 células [19].

Sea $\Sigma = \{0, 1\}$ el conjunto de estados, $\mathcal{V} = 26$ el número de vecinos, entonces una célula $c_{i,j,k} \in \mathbb{Z} \otimes \mathbb{Z} \otimes \mathbb{Z}$ es discretamente conectada a una célula $\{(c_{i+k_1}, c_{j+k_2}, c_{l+k_3}) : \text{Max}\{|k_1|, |k_2|, |k_3|\} \leq 1\}$, es decir, la vecindad de Moore en tres dimensiones, $c_0 = c_{i,j,k}$ es la célula central, mientras que $c_1, \dots, c_{\mathcal{V}} = c_{i-1,j-1,k-1}, \dots, c_{i+1,j+1,k+1}$ son los vecinos alrededor de la célula central para toda $c_i \in \Sigma$. En la ecuación que se muestra a continuación, la función ϕ define la transformación local, las variables N_{min} y S_{min} indican el número mínimo de células ocupadas por el estado 1 en \mathcal{V} en un tiempo t . Si $c_0 = 0$ en el tiempo t , entonces $c_0 = 1$ en el tiempo $t + 1$ si $N_{min} \leq \sum_{i=1}^{\mathcal{V}} c_i \leq N_{max}$. Si $c_0 = 1$ en el tiempo t , entonces $c_0 = 0$ en el tiempo $t + 1$ si $S_{min} \leq \sum_{i=1}^{\mathcal{V}} c_i \leq S_{max}$. Finalmente una regla semitotalística en tres dimensiones se representa como $R(S_{min}, S_{max}, N_{min}, N_{max})$, donde N y S deben tomar valores entre 1 y 26 Ec. 2.3.7.

$$\phi(c_0, c_1, \dots, c_{\mathcal{V}}) = \begin{cases} 1 & \text{si } \begin{cases} c_0 = 0 & \text{y} & N_{min} \leq \sum_{i=1}^{\mathcal{V}} c_i \leq N_{max} \\ c_0 = 1 & \text{y} & S_{min} \leq \sum_{i=1}^{\mathcal{V}} c_i \leq S_{max} \end{cases} \\ 0 & \text{otro caso} \end{cases} \quad (\text{Ec. 2.3.7})$$

Se puede observar que esta ecuación es la misma que (Ec. 2.3.1), vista en los AC en dos dimensiones, a diferencia de que para este caso como ya se mencionó N y S pueden tomar valores entre 1 y 26.

2.4. Autómata Celular complejo

2.4.1. Juego de la Vida (Game of Life)

¿Qué es?

“The Game Of Life” (o *Juego de la Vida*) es una invención de John Horton Conway, un matemático inglés, éste fue publicitado en varias ocasiones en la columna mensual de Martin Gardner llamada “*Mathematical Recreations*”.

Programadores en numerosos centros de cómputo exploraron el juego en sus máquinas teniendo como ventaja un lugar en el cual mostrar el juego, aun así, algunos otros solo utilizando lápiz y papel obtuvieron significantes resultados.

El juego de la vida es un tipo de AC en dos dimensiones, el cual está caracterizado por sus estados y reglas para decidir el cambio de los estados a través del tiempo. Como ya se sabe, un AC cambia sus estados simultáneamente de acuerdo a los estados de los vecinos en la red en la que se encuentra situado (entiéndase por red, la vecindad). Cada una de las células tiene solamente dos estados “0” o “1”; estos autómatas son los más simples. La distinción entre *vivos* y *muertos* fue dada en una versión de las publicaciones de *Life*.

Este AC tiene relación con la modelación de sistemas biológicos, tal como ecosistemas, incendios forestales, reacciones químicas, comportamientos colectivos de seres vivos, entre otros.

Reglas

John von Neumann eligió una red cuadrada de tal forma que cada célula tuviese 8 vecinos, es decir las células ortogonales y diagonales. La regla de evolución que eligió Conway fue la siguiente:

1. Una célula sobrevive si tiene dos o tres vecinos vivos
2. Una nueva célula nace en donde sea que existan tres vecinos vivos
3. El resto de células mueren o se mantienen inactivas

Mientras que el criterio de Conway fue que la regla utilizada para *Life* no debía permitir que la población pereciera rápidamente ni que se expandiera hasta sus límites.

Las características mencionadas anteriormente deben cumplir tres condiciones, cuando una célula debe *nacer*, *sobrevivir o morir*:

- **Nacimiento:** Una célula muerta en un tiempo t vive en un tiempo $t + 1$ si y solo si tres de sus vecinos están vivos en un tiempo t
- **Muerte por sobrepopulación:** Si una célula vive en un tiempo t y tiene cuatro o más de sus ocho vecinos vivos en un tiempo t , dicha célula debe morir en el tiempo $t + 1$
- **Muerte por aislamiento:** Si una está viva en el tiempo t y tiene menos de un vecino vivo en el tiempo t esta célula muere en el tiempo $t + 1$
- **Supervivencia:** Una célula que está viva en el tiempo t se mantiene viva en un tiempo $t + 1$ si y solo si tiene dos o tres vecinos vivos en el tiempo t

La función de transición ϕ emplea la vecindad de Moore, la regla de evolución es semitotalística y se representa como $R(2,3,3,3)$. De esta representación la pareja formada por 2,3 son un mínimo y un máximo respectivamente de células vivas para definir si la célula $c_0 = 1$ en un tiempo t sobrevive en un tiempo $t + 1$, la segunda pareja de números, es decir 3,3 son también un mínimo y un máximo respectivamente de células vivas para definir si la célula $c_0 = 0$ en un tiempo t nace en un tiempo $t + 1$, por tanto si se utiliza la ecuación (Ec. 2.3.1) las variables $S_{min} = 2, S_{max} = 3, N_{min} = 3, N_{max} = 3$, en consecuencia la ecuación (Ec. 2.3.1) cambia obteniendo Ec. 2.4.1

$$\phi(c_0, c_1, \dots, c_8) = \begin{cases} 1 & \text{si } \begin{cases} c_0 = 0 & \text{y} & 3 \leq \sum_{i=1}^8 c_i \leq 3 \\ c_0 = 1 & \text{y} & 2 \leq \sum_{i=1}^8 c_i \leq 3 \end{cases} \\ 0 & \text{otro caso} \end{cases} \quad (\text{Ec. 2.4.1})$$

2.5. Aplicaciones

Como se vio en el capítulo anterior un AC tiene la capacidad de modelar sistemas naturales o artificiales que puedan ser descritos como una colección de objetos simples que interactúan localmente unos con otros. Es por ello que los AC tienen aplicaciones en las diferentes ramas de la ciencia: ciencias formales, ciencias naturales e incluso en las ciencias sociales.

2.5.1. Arquitectura

Si se piensa en utilizar un AC en una o dos dimensiones para modelar construcciones quizás no tenga mucho sentido, en cambio sí se utilizan las tres dimensiones se pueden observar formas y patrones interesantes para la arquitectura.

La conexión con la arquitectura es la capacidad de los AC para generar patrones, a partir de patrones organizados se pueden sugerir nuevas formas arquitectónicas. Muchos métodos digitales en arquitectura son controlados paramétricamente, se utiliza un conjunto inicial de parámetros para generar un resultado. La diferencia entre el método paramétrico y el método recursivo es que en los primeros los resultados pueden anticiparse fácilmente, mientras que en estos últimos el resultado generalmente no puede.

Interpretación arquitectónica

La traducción matemática pura de un AC en forma arquitectónica incluye una serie de cuestiones que no consideran la realidad construida. En la Fig. 2.5.6 se muestra una configuración inicial y sus resultados en la octava generación y de lado derecho se encuentra la interpretación o traducción a una posible forma de construcción que se puede tratar después de que haya evolucionado o se puede considerar desde el principio.

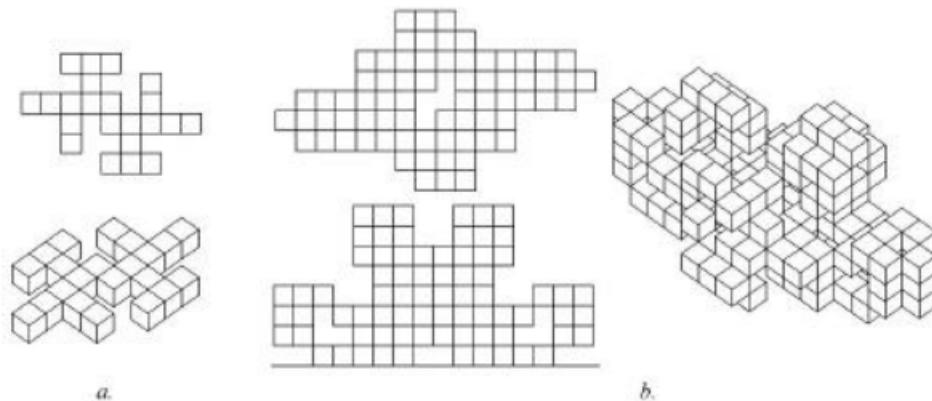


Fig. 2.5.6: Generación de muestra [20].

Fuente: Architectural Interpretation of Cellular Automata [20].

En Fig. 2.5.7 se puede observar otro ejemplo interesante.

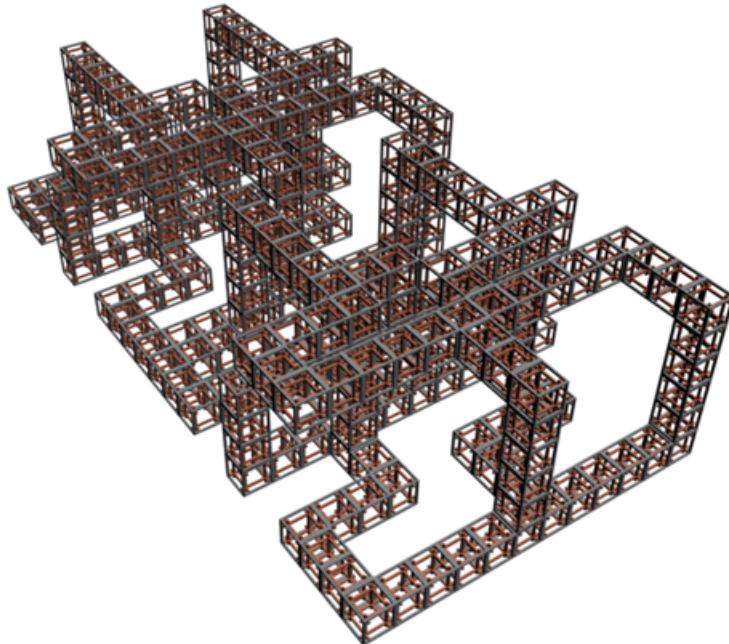


Fig. 2.5.7: GDesign 2.0 de Umberto Roncoroni [21].

Fuente: L-Systems Generator [21].

Este es un ejemplo de un software utilizado para arte, arquitectura y diseño: GDesign de Umberto Roncoroni. La aplicación trabaja una mixtura de “sistemas Lindenmayer” (L-Systems) y AC y el autor comparte los resultados obtenidos en su galería, dando la oportunidad de investigar más al respecto y considerar lo interesante de aplicar la teoría de los AC en esta rama.

2.5.2. Criptografía

La criptología (del griego *cripto*-culto- y *logos*-tratado, ciencia) se divide en dos ramas: criptografía, se ocupa de las técnicas de cifrado de información, de tal forma que dicha información solo sea inteligible para quien sepa descifrarlo y el criptoanálisis que se encarga de intentar obtener la información cifrada o la clave con que

fue cifrada mediante la información disponible[22].

El proceso de cifrar un mensaje consiste en transformarlo por medio de un algoritmo de tal forma que sea incomprendible por toda aquella persona que no posea la clave con la que fue transformado dicho mensaje. En síntesis, si el destinatario conoce esa clave podrá descifrar el mensaje y recuperar el mensaje original.

En el algoritmo se generan claves y un mensaje cifrado (conocido como *criptograma*); al proceso en conjunto se le llama *criptosistema* [22]. Con lo anterior, se puede entender la necesidad de obtener números aleatorios y pseudo-aleatorios para la generación de claves y que los mensajes puedan ser transmitidos de forma segura utilizando estas técnicas.

Stephen Wolfram fue el primero en proponer el uso de un AC unidimensional utilizando la regla 30 (este número está dado por el binario 00011110), debido a las propiedades de esta regla: no se observa simetría ni patrones repetidos, lo cual es señal de su aleatoriedad.

La idea es partir de una configuración inicial del autómata, que sería la semilla del generador, realizar algunas iteraciones y quedarse con los bits generados por una determinada célula (por ejemplo, la central) [23, 22].

Para este ejemplo se muestran 40 y 100 evoluciones de la regla 30 (Fig. 2.5.8) respectivamente que parten de una configuración inicial en la que todas las células se encuentran en estado 0 a excepción de la célula central que posee el estado 1, se puede observar que la célula central es efectivamente aleatoria, que no hay simetría ni patrones repetidos.

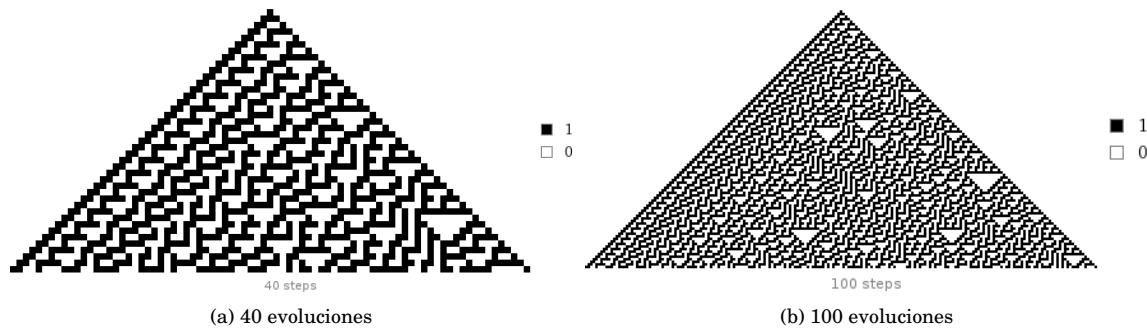


Fig. 2.5.8: Evolución de la regla 30 en un autómata unidimensional.

Fuente: Realización propia.

Con esto es posible generar secuencias simbólicas con alto grado de aleatoriedad [22]. El remitente produce un mensaje cifrado sumando los bits del mensaje original con una secuencia generada por el AC. Para que el destinatario pueda descifrar el mensaje debe generar una secuencia aleatoria a partir del mismo arreglo inicial y realizar una operación de suma al mensaje cifrado.

Cada caso es particular dependiendo de las características que se encuentran en las reglas y el uso que se les da, pero en el caso de la criptografía se ha visto la posibilidad de generar claves seguras que cumplen con su cometido haciendo que los mensajes tengan la propiedad de confidencialidad.

2.5.3. Modelado epidémico

Los problemas de salud pública son gran tema de discusión en los medios de comunicación debido a su impacto a nivel mundial. La muerte de personas por enfermedades como consecuencia de contraer enfermedades tiene un costo de millones de dólares en la industria anualmente, por tal razón es que se realizan constantemente investigaciones con el fin de predecir patrones de infecciones virales dadas ciertas condiciones ambientales [24].

La mayoría de los modelos actuales hacen uso de ecuaciones diferenciales y no toman en cuenta algunos factores parciales como la densidad de la población y la dinámica de la población.

Los modelos epidémicos son bastante complicados ya que es difícil incorporar todas las variables existentes en la naturaleza por esa razón en una simulación es necesario acotar y precisar las variables a utilizar.

En el marco de representación epidémica (SIR) se explica que un anfitrión puede tener alguno de los siguientes

estados: susceptible, infeccioso o recuperado. Los anfitriones con el estado de susceptible son aquellos que están sanos y no han sido contagiados con el virus, pero es posible que lo contraigan de algún anfitrión infectado. Los anfitriones que han sido infectados tienen posibilidades de contagiar a otro anfitrión o bien no hacerlo. Finalmente, los que poseen el estado de recuperados son aquellos que no poseen el virus y además han adquirido inmunidad a futuras infecciones.

Shih Ching Fu y George Milne desarrollaron un software el cual permite simular epidemias. Este simulador requiere que sean ingresados diversos parámetros algunos de ellos incluso probabilísticos:

- Radio de infección: determina el tamaño de la vecindad de interacción de una célula, el modelo utilizado es del tipo Moore cuya área es de n y la interacción es determinada por el radio r el área es decir n se calcula con la ecuación:

$$n = (2r + 1)^2$$

- Tasa de inmigración
- Tasa de nacimiento
- Tasa de mortalidad por causas naturales
- Morbilidad viral: problemas médicos causados por algún tratamiento
- Tasa espontánea de infección
- Tasa vectorial de infección: un virus puede contagiarse de huésped a huésped, pero también es posible que sea transmitido utilizando sus medios de propagación naturales. Para este modelo se hace uso de la siguiente ecuación para la obtención de este parámetro:

$$P_v = \frac{\text{Población susceptible}}{\text{capacidad de la vecindad}} \cdot P_i$$

donde:

- P_v : es la función de propagación del vector
- P_i : es el parámetro de infección y densidad susceptible

- Tasa de infección por contacto
- Tasa de recuperación
- Tasa de resusceptibilidad
- Probabilidad de movimiento: limitante para definir áreas como océanos límites fronterizos etc.

Este software permite realizar simulaciones en diversos escenarios y muestra que el uso de AC tiene un gran impacto también en esta área, ya que hay posibilidad de llegar a conclusiones epidemiológicas [24].

Con las aplicaciones revisadas en esta sección se puede observar parte de la importancia del estudio de los AC, por lo que resulta de gran interés conocerlos a detalle y poder realizar sus propios modelos para diversas disciplinas para resolver determinados problemas.

2.6. Teoría del campo promedio

La teoría del campo promedio es un modelo simple que ha sido utilizado principalmente para caracterizar el espacio completo de las reglas de los autómatas celulares mediante propiedades estadísticas de los mismos [17, 19, 25].

Esta teoría describe cómo actúan los AC sobre medidas de probabilidad de un tipo particular. Una medida de probabilidad es la asignación de una probabilidad a vecindades de células de todos los tamaños [25].

Si se desea calcular la probabilidad de encontrar una vecindad de células en la siguiente generación, se puede estimar mediante su frecuencia de aparición en generaciones ancestrales. Por definición, la probabilidad de aparición de que una célula b tenga un determinado estado en el tiempo $t + 1$, es la suma de probabilidades de que ocurran las vecindades B que toman ese estado en el tiempo t [25].

Formalmente el cálculo de la probabilidad de encontrar un determinado estado en el tiempo $t + 1$ está dado por la Ec. 2.6.1

$$p_{t+1} = \sum_{j=1}^{k^{2r+1}} \phi_j(X) P_t^v (1-p_t)^{n-v} \quad (\text{Ec. 2.6.1})$$

donde

- ϕ : Regla de evolución de un AC de orden (k, r) donde cada célula tiene dos estados, $\Sigma = \{0, 1\}$
 - Σ es el conjunto de estados independientes entre sí, es decir, no hay ningún tipo de relación entre los elementos del espacio de evoluciones
- La vecindad es de tamaño $2r + 1$
- k^{2r+1} es el número de vecindades
- p es la probabilidad de tener un estado 1 en el tiempo t
- $1 - p$ es la probabilidad de tener el estado 0 en el tiempo t
- v es el número de veces que aparece el estado 1 en la vecindad
- $n - v$ es el número de veces que aparece el estado 0 en la vecindad
- La suma toma los valores de todas las vecindades definidas por el conjunto Σ^{2r+1}

Cabe destacar que la teoría del campo promedio puede ser aplicada a AC de diferentes dimensiones; por ejemplo en investigaciones como las de Howard Gutowitz [25] que se enfoca principalmente en los AC unidimensionales y en este trabajo se estará utilizando para realizar el análisis en espacios binarios bidimensionales, es decir, $k = 2$, ya que cada celda puede tomar valores de 0 o 1.

2.7. Clases de puntos fijos

Los sistemas dinámicos poseen propiedades asintóticas de las soluciones y las trayectorias. El tipo de comportamiento más simple se presenta en los puntos de equilibrio o también llamados puntos fijos y las órbitas periódicas. Es necesario preguntarnos si modificando un poco la configuración inicial con el paso del tiempo el sistema tendrá un comportamiento similar o bien converge a un punto en particular. En caso de que el sistema oscile en valores cercanos a algún punto este se le llama del tipo **estable** pero si converge a cierto punto se le llama asintóticamente **estable** y es del tipo atrayente.

Cuando hablamos de estabilidad nos referimos a que si perturbamos un poco el sistema este no sufre grandes cambios pero tenemos el caso opuesto también es decir si perturbamos el sistema un poco este no logra estabilizarse estos puntos también son de interés.

Se pueden realizar diversas pruebas para analizar la estabilidad de un sistema que es no lineal. Consideremos lo siguiente:

Sea $f : R \rightarrow R$ una función continua diferenciable con un punto fijo a , $f(a) = a$. Consideremos que el sistema dinámico es obtenido iterando la función f es decir:

$$x_{k+1} = f(x_k), k = 0, 1, 2, \dots, n$$

El punto fijo a se le llama **estable** si el valor absoluto de la derivada de f en a es estrictamente menor que 1, por otra parte se le llama **inestable** si este valor es estrictamente mayor a 1. Esto ocurre porque la función f tiene una aproximación lineal con pendiente $f'(a)$ es decir:

$$f(x) \approx f(a) + f'(a)(x - a)$$

Entonces

$$x_{k+1} - x_k = f(x_k) - x_k \approx f(a) + f'(a)(x_k - a) - x_k = a + f'(a)(x_k - a) - x_k = (f'(a) - 1)(x_k - a) \rightarrow \frac{x_{k+1} - x_k}{x_k - a} = f'(a) - 1$$

Lo anterior significa que la derivada mide la velocidad a la que las iteraciones sucesivas se acercan al punto fijo, o en caso contrario divergen de dicho punto. Si el resultante de lo anterior es 1 o -1 se requiere más información para decidir la estabilidad del sistema, en dicho caso llamaremos a esos puntos como **indiferentes** [27].

2.8. Entropía

Otra herramienta muy útil para realizar el análisis del comportamiento de las reglas y su clasificación es el cálculo de la entropía.

La entropía es el segundo principio de la termodinámica que se encarga del estudio de la evolución de los sistemas termodinámicos; etimológicamente la palabra entropía viene del griego *em* que significa sobre, en, cerca de, y *sqopg* que significa giro, alternativa, cambio y evolución. Este término lo acuñó el físico Rudolf Julius Emmanuel Clausius en el año 1850 [28, 29]. Más tarde se realizaron analogías con el comportamiento de moléculas (biología), en comunicaciones y en diferentes ramas de la ciencia en general.

Actualmente existe una estrecha relación entre la entropía y la teoría de la información; específicamente para esta rama de la ciencia se le conoce como *entropía de Shannon*.

Esta entropía está dada por la Ec. 2.8.1

$$H(X) = - \sum_{x \in X} p_x \log_2 p_x \quad (\text{Ec. 2.8.1})$$

donde

- p_x es la probabilidad de que ocurra un mensaje
- x es un mensaje
- X es un conjunto de mensajes
- el signo de resta (-) asegura que las entropías sean cantidades positivas
- H promedia el número de opciones binarias necesarias para seleccionar un mensaje de un conjunto más grande, o el número de dígitos binarios (bits) necesarios para enumerar ese conjunto.

Si bien la base de este logaritmo es arbitraria, Shannon la estableció en dos, por lo que reconoce que la elección entre dos alternativas equiprobables es la opción más elemental conceible [30]. La entropía para este caso en particular donde hay dos alternativas con las probabilidades p y $q = 1$, queda denotada como Ec. 2.8.2

$$H = -(p \log p + q \log q) \quad (\text{Ec. 2.8.2})$$

de la cual se muestra su gráfica en Fig. 2.8.9 como una función de p .

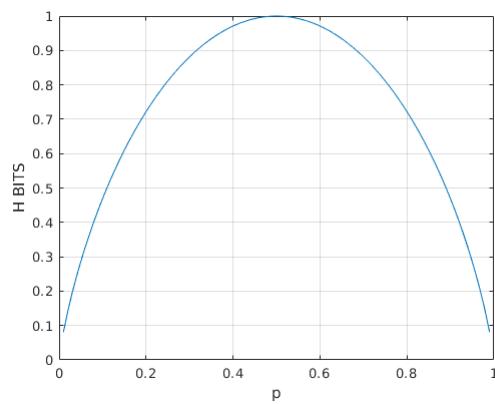


Fig. 2.8.9: Entropía para el caso de dos alternativas con probabilidades p y $(1-p)$
Fuente: "A Mathematical Theory of Communication" [31].

La cantidad de H tiene propiedades muy interesantes que justifican el cálculo de esta entropía como una medida razonable de elección o información [31].

1. $H = 0$ si y solo si todas las probabilidades p_i excepto una (que tenga el valor de la unidad) son cero.
2. Dado n , H es un máximo e igual a $\log n$ cuando todas las probabilidades p_i son iguales, es decir, $\frac{1}{n}$.
3. Para dos eventos, x e y con m y n posibilidades respectivamente; sea $p(i,j)$ la probabilidad de ocurrencia conjunta de i para el primero y j para el segundo, la entropía del evento conjunción es

$$H(x,y) = - \sum_{i,j} p(i,j) \log p(i,j) \quad (\text{Ec. 2.8.3})$$

4. Cualquier cambio hacia la igualación de las probabilidades p_1, p_2, \dots, p_n aumenta H , por lo tanto si $p_1 < p_2$ y se aumenta p_1 , entonces p_2 se disminuye en una cantidad igual para que ambas probabilidades sean más o menos iguales provocando que H aumente.

Como se puede intuir, esta entropía ayudará a determinar la probabilidad de ocurrencia del estado de una célula en un tiempo t , ya sea que esta tome un valor de 0 o de 1. La entropía se puede utilizar para estudiar la cantidad de información en la evolución de un AC y ayuda a realizar la clasificación de reglas que se pretende.

En la siguiente tabla se muestra el cálculo de la entropía de The Game Of Life. Para realizar dicho cálculo, se ha empleado (Ec. 2.8.2), siendo que p representa la probabilidad de aparición de una célula viva en la actual generación y q la probabilidad de aparición de una célula muerta.

Generación	p	q	Entropía
1	0.2962	0.7038	0.8765
10	0.2068	0.7932	0.7353
50	0.1093	0.8907	0.4978
100	0.0918	0.9082	0.4424
200	0.0565	0.9435	0.3133
400	0.0516	0.9484	0.2931
600	0.0380	0.9620	0.2330
800	0.0386	0.9614	0.2358
1000	0.0305	0.9695	0.1968

2.9. Sistemas similares

Como parte del estudio es necesario conocer los trabajos y sistemas que se han realizado anteriormente por otros autores con la finalidad de reconocer y aportar nuevas funcionalidades principalmente en el área de investigación, análisis y clasificación de reglas en AC de dos dimensiones que utilicen la vecindad de von Neumann, por ello a continuación se presentan algunos de estos trabajos publicados y sus características haciendo énfasis en las ventajas y desventajas que presenta comparándolos con el simulador y la exploración que se pretende realizar.

2.9.1. Golly

Golly es una aplicación multiplataforma de código abierto para explorar “*The Game of Life*” de John Conway y muchos otros tipos de AC. Los autores principales son Andrew Trevorrow y Tom Rokicki, con contribuciones de código de Tim Hutton, Dave Greene, Jason Summers, Maks Verver, Robert Munafo, Brenton Bostick y Chris Rowett.

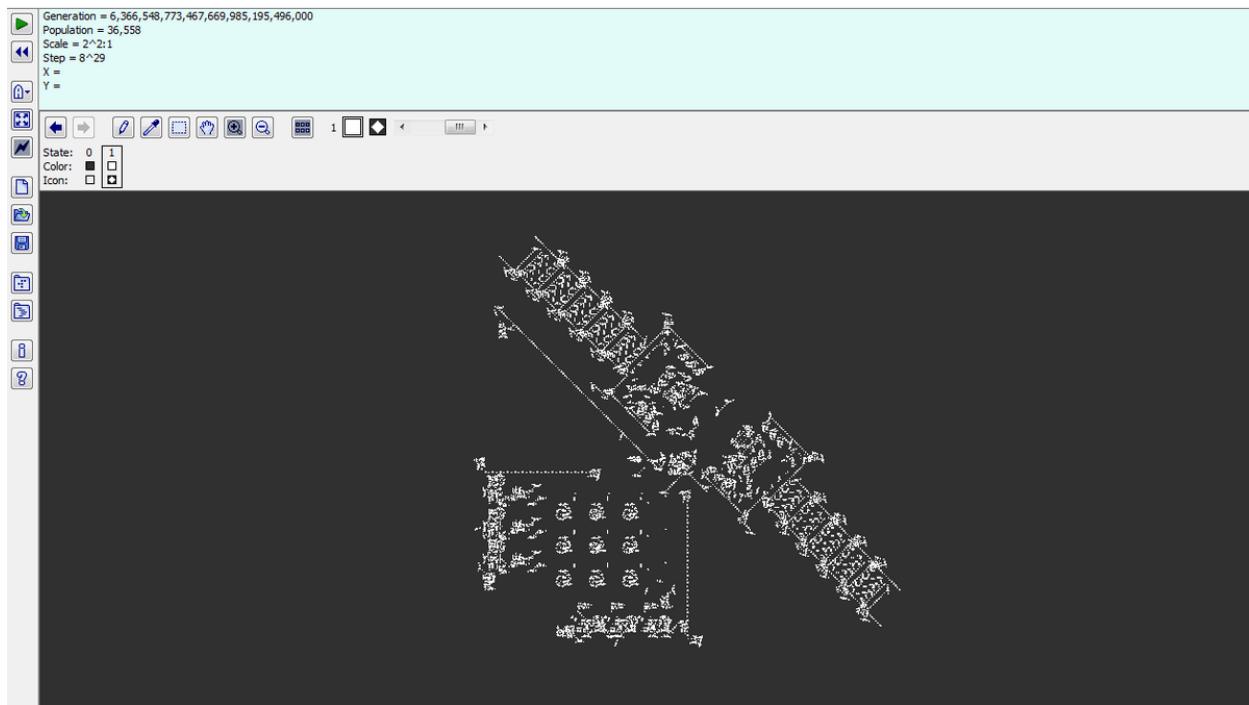


Fig. 2.9.10: Máquina de Turing simulada en Golly.

Fuente: Realización propia.

La aplicación está escrita en el lenguaje de programación *C++* empleando la librería *WxWidgets*, esta librería permite un desarrollo más sencillo de aplicaciones gráficas. Golly además, emplea el potente algoritmo *Hashlife* que puede alcanzar una cifra muy grande de generaciones en un breve periodo de tiempo.

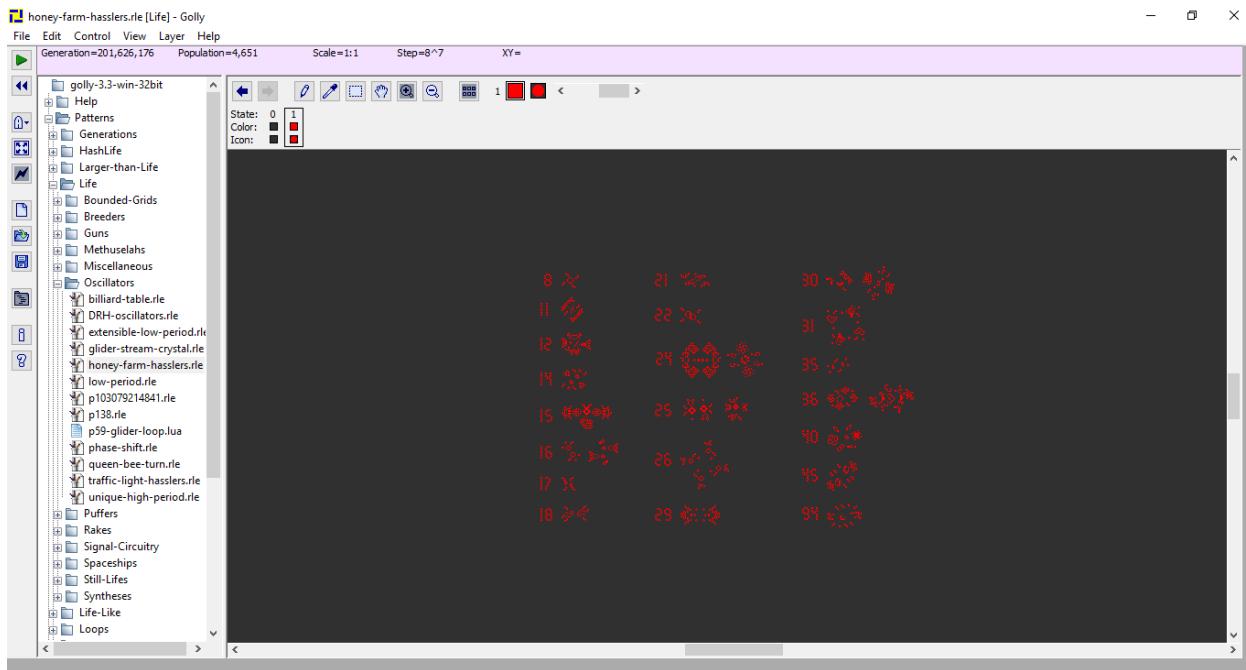


Fig. 2.9.11: Oscilador alcanzando la generación número 201,626,176 en menos de 1 minuto.
Fuente: Realización propia.

En Fig. 2.9.10 se muestra una simulación utilizando el software Golly de una máquina de Turing y en Fig. 2.9.11 se muestra la simulación de un oscilador.

Características

- Gratuito y de código libre (Open Source)
- Admite universos acotados y no acotados, con células de hasta 256 estados
- Admite múltiples algoritmos, incluido el algoritmo súper rápido Hashlife de Bill Gosper
- Incluye diferentes tipos de AC: John von Neumann's 29-state CA, Wolfram's 1D rules, WireWorld, Generations, Paterson's Worms, Larger than Life, etc
- Admite “cajas ocupadas” y otras reglas de AC en 3D
- El algoritmo RuleLoader le permite conectar nuevas reglas
- Responsividad incluso mientras genera o recolecta basura
- Lee archivos RLE, macrocell, Life 1.05 / 1.06, dblife y MCell
- Lee formatos gráficos comunes: BMP, PNG, GIF, TIFF
- Permite extraer patrones, reglas y scripts de archivos “.zip”
- Permite descargar patrones, reglas y scripts de archivos en línea
- Incluye una colección de patrones
- Permite pegar patrones desde el portapapeles
- Historial de versiones ilimitado (deshacer / rehacer)

- Atajos de teclado configurables
 - La opción de ajuste automático mantiene los patrones ajustados a la ventana
 - Opción de pantalla completa (sin menú / estado / herramienta / barras de desplazamiento)
 - Admite múltiples capas, incluidas las capas clonadas
 - Ayuda basada en HTML con un Life Lexicon integrado
 - Se pueden hacer scripts por medio de Lua o Python
 - Se ejecuta en Windows (XP +), Mac OS X (10.6+) y Linux (con GTK + 2.x)

Este software se puede descargar a través de su página en SourceForge, donde también se encuentran las características más importantes del mismo.

Comparativa con el simulador propuesto en este trabajo (vNCASimulator)

Con estas características se puede observar que “Golly” es un simulador de AC muy completo que a pesar de ser un software demasiado robusto presenta la desventaja de que no es muy intuitivo para usuarios que no están familiarizados con el software o que son inexpertos en el tema esto puede observarse en Fig. 2.9.14.

Con el simulador vNCASimulator se pretende que tenga una interfaz gráfica sencilla para que el usuario pueda establecer el tipo de vecindad, las reglas de nacimiento y supervivencia y toda la configuración inicial deseada.

La ventaja más notable que el simulador vNCASimulator presenta sobre “Golly” es el uso de la teoría de campo promedio para la obtención del polinomio de las diferentes reglas, la gráfica del polinomio, la gráfica en tiempo real de la población del AC y la gráfica de la entropía.

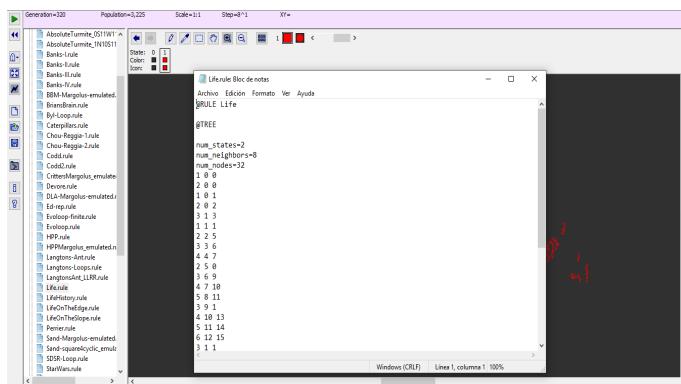


Fig. 2.9.12: Archivo de reglas en Golly

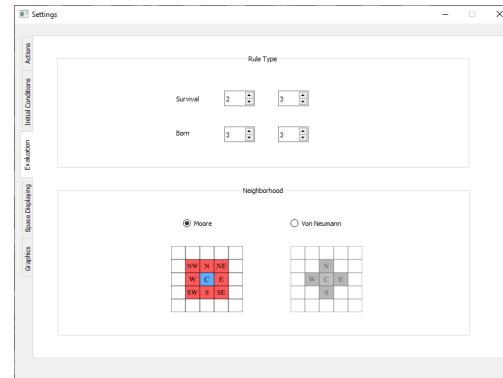


Fig. 2.9.13: Reglas en el simulador vNCASImulator

Fig. 2.9.14: Comparativa de software Golly y vNCASimulator

Fuente: Realización propia.

2.9.2. ACOSXL21, ACOSXSTV, ACOSXSTM, ACOSXV y ACOSXW

Elaborados por el Dr. Genaro Juárez Martínez para su tesis de maestría titulada “Teoría del Campo Promedio en Similares a The Gameof Life” [19], estos 5 programas se encuentran escritos en el lenguaje de programación orientado a objetos **Objective-C**, implementados inicialmente en el sistema operativo OpenStep y posteriormente para los sistemas *Mac OS X “Aqua”*, *Mac OS X Server “Rhapsody”* y *Windows NT*.

ACOSXL21

ACOSXL21 (Fig. 2.9.15) es un simulador de AC en una dimensión de orden $k = 2$ y $r = 1$, permite variar el tamaño de las células, establecer configuraciones aleatorias de diferentes densidades, continuar la evolución si se desea y limpiar el espacio de evoluciones.

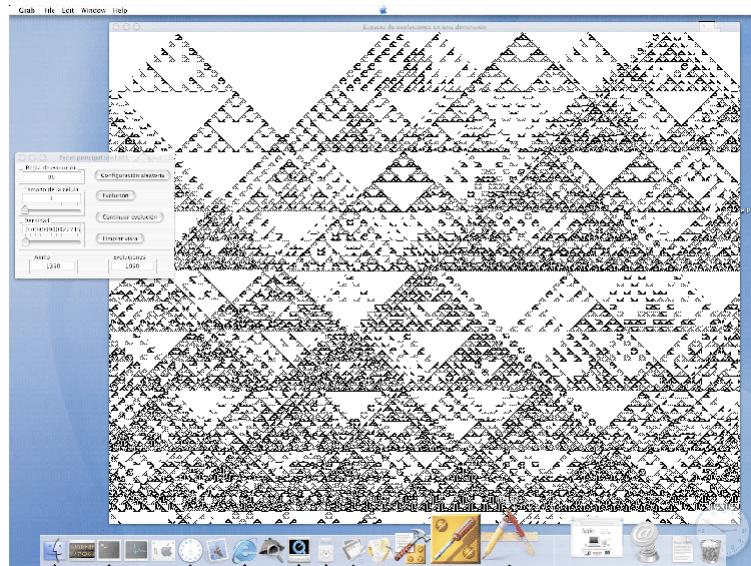


Fig. 2.9.15: ACOSXL21 para Mac OS X [32]

Fuente: Realización propia.

ACOSXSTV

ACOSXSTV (Fig. 2.9.16) es un simulador de AC en dos dimensiones que trabaja con la vecindad de von Neumann y emplea reglas semitotalísticas, la regla de evolución funciona con enteros dentro del rango 1 a 4, se puede cambiar el estado de la célula al presionar sobre ella en el espacio de evolución, calcular la siguiente generación, evolucionar de manera continua y detenerlo con el mismo botón, limpiar el espacio de evolución, así como introducir una evolución del tipo de la regla 110.

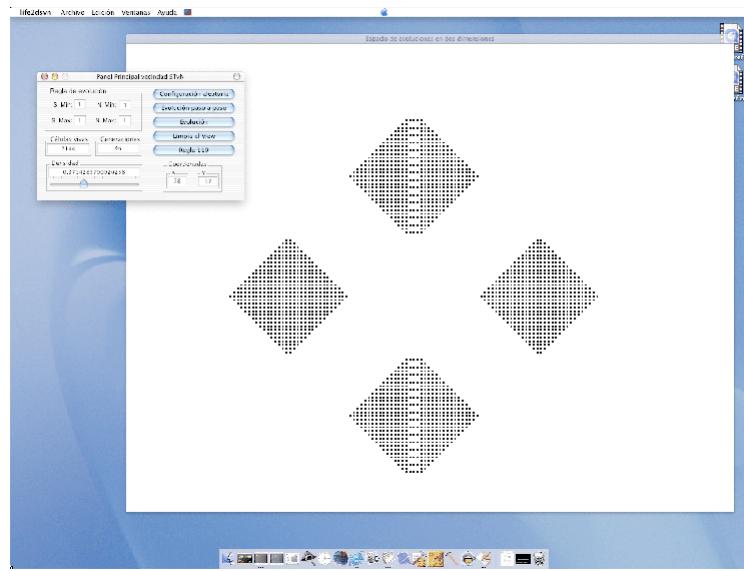


Fig. 2.9.16: ACOSXSTV para Mac OS X [33]

Fuente: Realización propia.

ACOSXSTM

ACOSXSTM (Fig. 2.9.17) es un simulador de AC en dos dimensiones que trabaja con la vecindad de Moore y emplea reglas semitotalísticas, la regla de evolución puede variar con enteros dentro del rango 1 a 8, pude establecer configuraciones aleatorias de diferentes densidades, permite cambiar el estado de la célula al presionar sobre ella dentro del espacio de evolución, calcular la siguiente generación y evolucionar de manera continua, detenerla en el momento que se desee pulsando el mismo botón, limpiar el espacio de evolución, así como introducir una evolución del tipo de la regla 110.

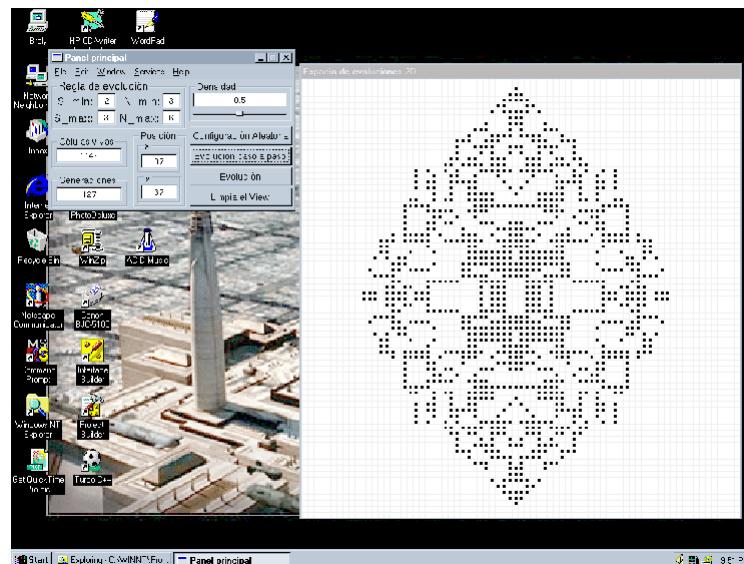


Fig. 2.9.17: ACOSXSTM para Windows NT ACOSXSTM [34]

Fuente: Realización propia.

ACOSXV

ACOSXV (Fig. 2.9.18) es un simulador de AC en dos dimensiones que trabaja con la vecindad de von Neumann completa, es decir, se pueden manejar las 32 vecindades que se derivan de esta vecindad, la regla de evolución cambia su estado únicamente presionando los botones, se pueden introducir reglas aleatorias, limpiar el arreglo de los botones, establecer configuraciones aleatorias de diferentes densidades, introducir células dentro del espacio de evoluciones, calcular las siguientes generaciones de manera continua y detener la evolución en el momento que se desee presionando el mismo botón, limpiar el espacio de evoluciones, así como introducir una evolución del tipo de la regla 110.

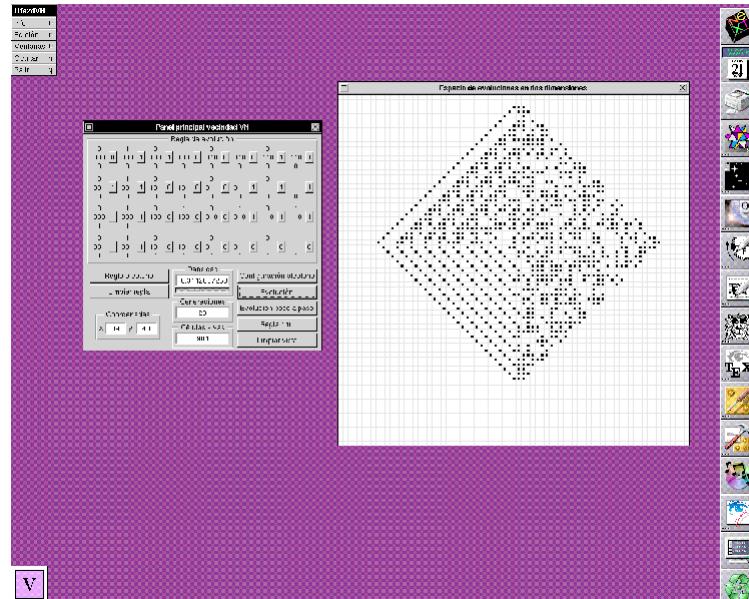


Fig. 2.9.18: ACOSXV para OpenStep [35]

Fuente: Realización propia.

ACOSXM

ACOSXM (Fig. 2.9.19) es un simulador de AC en dos dimensiones que trabaja con la vecindad de Moore completa, es decir, se pueden manejar las 512 vecindades que se derivan de esta vecindad, la regla de evolución cambia su estado únicamente dando presionando los botones y se muestra la vecindad que es activada, se pueden introducir reglas aleatorias, limpiar el arreglo de los botones, establecer configuraciones aleatorias de diferentes densidades, introducir con el ratón células dentro del espacio de evoluciones, calcular las siguientes generaciones de manera continua y detener la evolución en el momento que se desee pulsando el mismo botón, evolucionar paso a paso, limpiar el espacio de evoluciones, así como introducir una evolución del tipo de la regla 110.

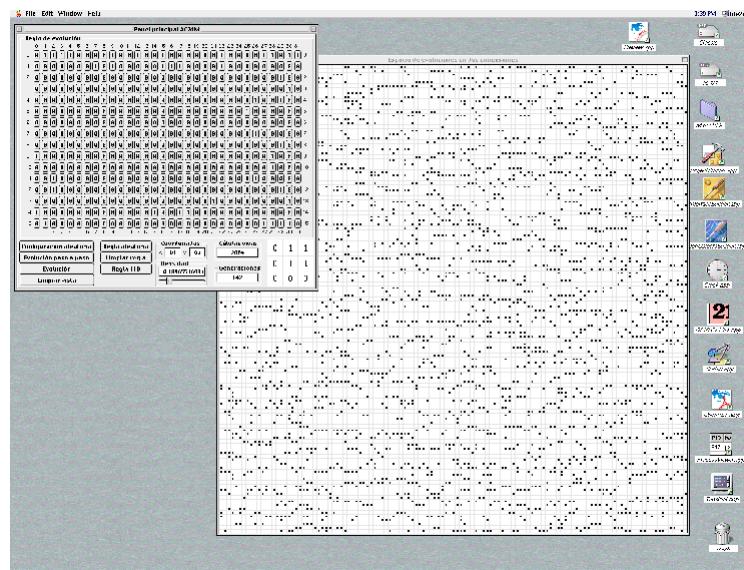


Fig. 2.9.19: ACOSXM para Mac OS X Server [36]

Fuente: Realización propia.

Comparativa con el simulador propuesto en este trabajo (vNCASimulator)

Este conjunto de programas presenta una similitud frente al software propuesto con una ligera excepción de ACOSXL21, ya que este funciona con AC en una dimensión.

Mientras que ACOSXSTV y ACOSXSTM son simuladores de von Neumann con reglas semitotalísticas y de Moore con reglas semitotalísticas, ACOSXV y ACOSXM funcionan con vecindad completa de Moore y de von Neumann.

El software proporciona al usuario la posibilidad de establecer la regla de evaluación a través de un arreglo de botones (32 botones para el caso de ACOSXV y 512 botones para ACOSXM), por otro lado, el software que se propone no evalúa la vecindad de Moore completa.

La característica más importante a resaltar del simulador vNCASimulator es la obtención del polinomio característico mediante la aplicación de la teoría del campo promedio, así como su gráfica esto lo podemos observar en Fig. 2.9.20 donde se muestra el polinomio característico de The Game Of Life.

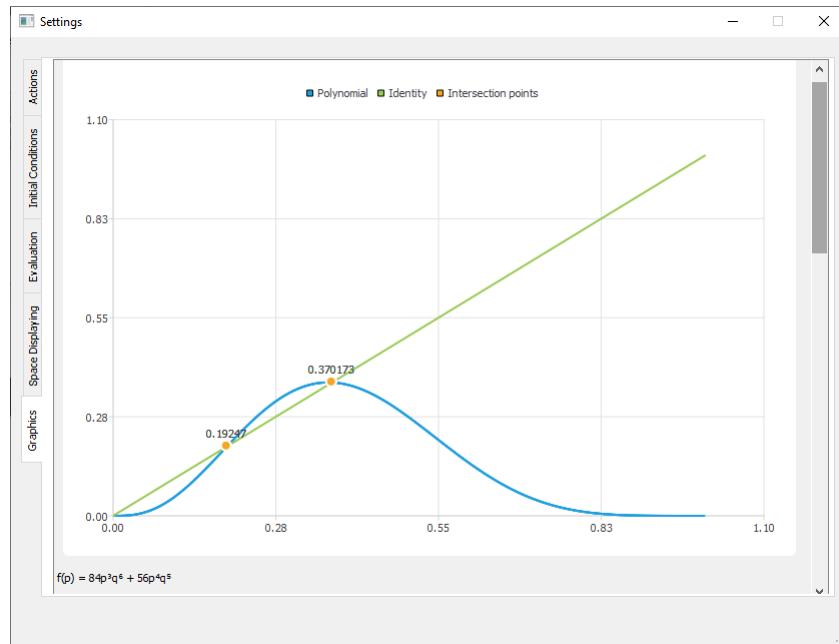


Fig. 2.9.20: Polinomio de campo promedio de Life por el software vNCASimulator

Fuente: Realización propia.

Otras características del simulador vNCASimulator son la capacidad de manejar el espacio de evolución, puede ser abierto o cerrado, que el usuario pueda visualizar una proyección a tres dimensiones de las células como se muestra en Fig. 2.9.21 y exportar e importar la configuración del AC a través de un archivo binario.

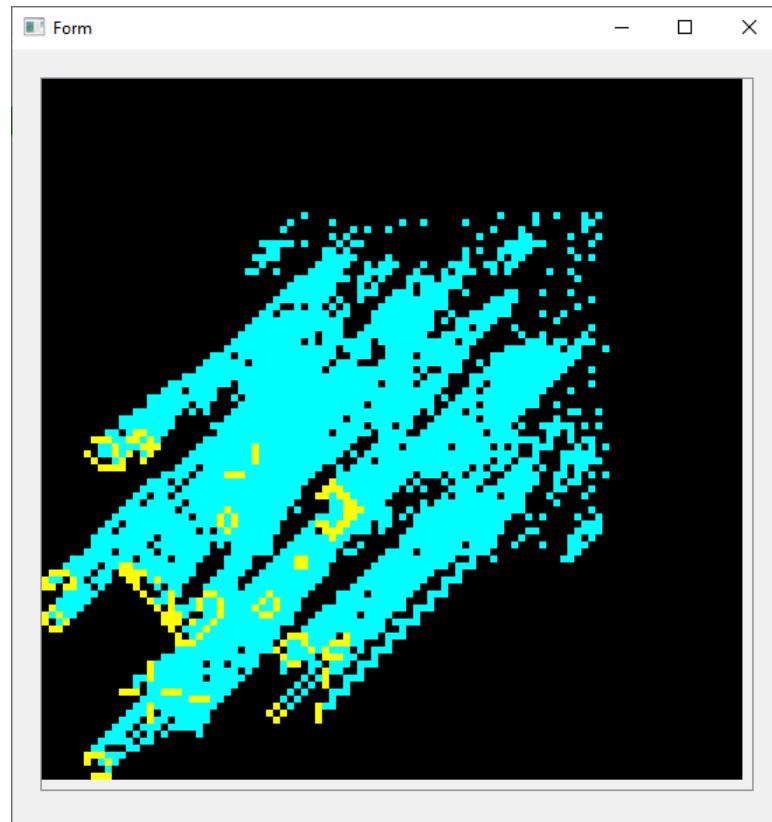


Fig. 2.9.21: Proyección en tres dimensiones

Fuente: Realización propia.

2.9.3. samcodes.co

Game of Life Cellular Automaton
The Game of Life, a simulation devised by John Conway

THE GAME OF LIFE

Interact with Conway's Game of Life in this [open source](#) implementation, written in [Haxe](#) by [Sam Twidale](#). Featuring patterns from [LifeWiki](#). Tap the canvas below to add Life patterns to the simulation.

Fig. 2.9.22: Sitio web www.samcodes.co.uk

Fuente: Realización propia.

El sitio web www.samcodes.co.uk (Fig. 2.9.22) ofrece un simulador de AC gratuito y de código libre. Este simulador fue desarrollado por Samuel Twidale, se encuentra escrito en **Haxe**, el cual es un lenguaje de programación estrictamente tipado, de alto nivel y código abierto con un compilador cruzado de optimización rápida. Esta implementación tiene precargados algunos patrones que se encuentran disponibles en el sitio web www.conwaylife.com, donde el usuario puede seleccionar alguno de éstos y colocarlo en el espacio de evolución esto se puede ver en Fig. 2.9.23.

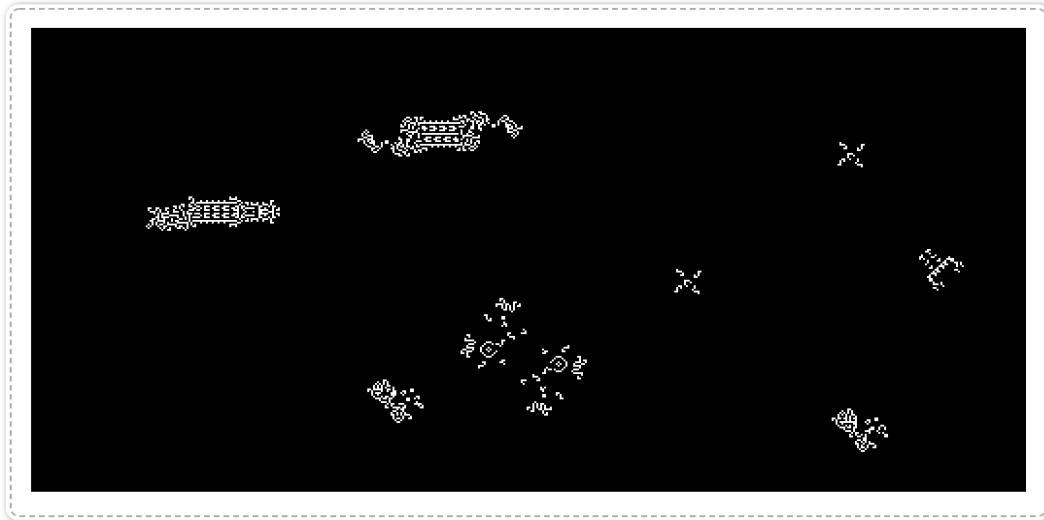


Fig. 2.9.23: Distintos patrones disponibles

Fuente: Realización propia.

Comparativa con el simulador propuesto en este trabajo (vNCASimulator)

Existen distintas implementaciones web de simuladores de AC, algunas de éstas emplean el algoritmo Hashlife de Bill Gosper, otras emplean ingeniosas técnicas de renderizado como la implementación de Samuel, que utiliza una técnica llamada **ping-pong**. Muchas de estas implementaciones que se encuentran en páginas web incluyendo la de Samuel no permite cambiar la regla con la que se requiere trabajar, tampoco permite cambiar el tipo de vecindad, únicamente permiten interactuar con el espacio de evolución por medio de patrones definidos.

Esta implementación tampoco proporciona gráficas en tiempo real del polinomio de campo promedio, población a través de las generaciones ni la entropía.

Parte III

Análisis y Diseño

3

Introducción a la metodología

Dado que en este caso el simulador de AC es solo una herramienta que permitirá el estudio del comportamiento de los mismos, se decidió utilizar una metodología que tiene como prioridad el desarrollo de prototipos funcionales.

La metodología es **RAD**, por sus siglas en inglés (*Rapid Application Development*), es decir, *Desarrollo Rápido de Aplicaciones*. A diferencia de otras metodologías como la de cascada que son comúnmente utilizadas para el desarrollo del software, esta prioriza el uso de librerías que ya existen y que han sido probadas para enfocarse en las demás funcionalidades que no han sido programadas ni probadas. No por ello se debe entender que al hacer uso de las librerías, estas no deben ser puestas a prueba, pero se ahorrará tiempo en dicha etapa. Para utilizar esta metodología es necesario conocer sus fases:

Fase 0: Planificación de requerimientos: En esta etapa, los desarrolladores, clientes y miembros del equipo deberán determinar los objetivos y resultados del proyecto, así como la discusión del problema y los posibles problemas a los que se han de enfrentar a lo largo del desarrollo.

Fase 1: Interfaz del usuario: Una vez que el proyecto esté definido se pasa directamente al desarrollo iniciando por el diseño de interfaces a través de varias iteraciones de prototipo.

Fase 2: Desarrollo: Esta etapa toma el diseño de los prototipos realizados en la fase anterior y lo convierte en un prototipo funcional.

Fase 3: Implementación: En esta última fase hay un producto terminado que deberá ser probado, tener la última actualización y se realiza la capacitación de usuarios.

Todos los cambios finales se realizan mientras los desarrolladores y los clientes continúan buscando errores en el sistema [37, 38].

4

Fase 1: Planificación de requerimientos

Para obtener los requerimientos primeramente se hace una descripción detallada del problema.

4.1. Problemática

En la actualidad los AC con vecindades del tipo von Neumann no han sido ampliamente estudiados, por lo tanto no existe una clasificación que permita agruparlos de acuerdo a sus características.

4.1.1. Planteamiento de solución

En este trabajo se propone realizar una exploración de diferentes reglas en los AC bidimensionales, específicamente en *Game Of Life* bajo la vecindad de von Neumann para observar patrones y su comportamiento de acuerdo diferentes condiciones iniciales, así como la obtención de los polinomios característicos que permitan asociarlos con las reglas en cuestión y para facilitar dicha exploración se propone desarrollar un sistema a la medida que permita modificar las configuraciones iniciales del AC, así como observar la evolución del espacio a través del tiempo y clasificar los puntos fijos obtenidos del polinomio en un entorno multiplataforma, es decir, que permita trabajar en diferentes sistemas operativos como Linux, Mac OS y Windows.

4.2. Requerimientos

Para el desarrollo del sistema se han establecido ciertos requerimientos mínimos tanto para software como para hardware que son necesarios para poder dar soporte, mantenimiento y/o utilizar el sistema.

4.2.1. Requerimientos de hardware

A nivel hardware se encuentran las características mínimas (físicas) que se listan a continuación:

- Equipo de cómputo (de escritorio, laptop o notebook)
- Procesador 1.4GHz
- Memoria RAM: 2 GB
- Disco Duro: 10 GB

4.2.2. Requerimientos de software

A nivel software se encuentran las características mínimas (intangibles) para poder dar soporte y/o utilizar el sistema como el IDE (entorno de desarrollo integrado), sistema operativo, entre otros.

- Sistema operativo: Linux 4.5 o superior, Windows 7 o superior, Mac OS X 10.0 o superior de 64 bits
- GCC 10.2.0 o superior
- Qt 5.15 o superior
- Qt Creator 4.13 o superior

Así mismo, para la exploración se determinó desarrollar un simulador de AC bidimensionales hecho a la medida con proyección al espacio tridimensional que permita observar el comportamiento de las células de forma visual en el transcurso del tiempo. Con esto se pudieron obtener y proponer los requerimientos para el desarrollo del simulador que se detallan a continuación.

4.2.3. Requerimientos funcionales

Los requerimientos funcionales son declaraciones de los servicios que debe proporcionar el sistema [39], en este caso el simulador, de manera que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares.

Requerimiento	Nombre	Descripción
RF01	Polinomio característico	Calcular el polinomio característico utilizando teoría del campo promedio
RF02	Siguiente generación	Calcular la siguiente generación del AC con base en las reglas y tipo de vecindad definidas y desplegarla en el espacio de evolución
RF03	Modificar célula	Permitir el cambio de estado para cada célula
RF04	Entropía	Calcular la entropía de Shannon con base en la probabilidad de aparición de una célula viva en la n-ésima generación
RF05	Evaluación temporal	El sistema debe ser capaz de comenzar, detener y reanudar la simulación a petición del usuario
RF06	Nuevo espacio de evaluación	Mostrar un nuevo espacio de evaluación con la configuración inicial proporcionada por el usuario
RF07	Espacio con células aleatorias	Permitir que el espacio se llene de forma aleatoria con base en una densidad proporcionada por el usuario
RF08	Proyección a tres dimensiones	Proyectar la simulación del autómata bidimensional a tres dimensiones
RF09	Tipo de vecindad	Permitir la selección de la vecindad (de von Neumann o de Moore) para realizar la evaluación sobre las células
RF10	Gráfica del polinomio característico	Generar gráfica del polinomio característico para la regla seleccionada comparado con la recta identidad en un rango de 0 a 1
RF11	Gráfica de la población	Generar gráfica de la densidad de población en el transcurso del tiempo
RF12	Tamaño de células	Permitir modificar el tamaño de las células
RF13	Gráfica de la entropía	Generar gráfica de la entropía en el transcurso del tiempo
RF14	Datos de la generación	Mostrar el número de generación, densidad de población y la entropía para la simulación en curso

Requerimiento	Nombre	Descripción
RF15	Velocidad de simulación	Permitir la modificación de la velocidad para mostrar la siguiente generación
RF16	Dimensión del espacio	Permitir la modificación de la altura y anchura del espacio de evolución
RF17	Puntos de intersección	Calcular y mostrar los puntos de intersección entre el polinomio característico y la recta identidad
RF18	Reglas	Permitir la modificación de las reglas para la evaluación de las células
RF19	Tipo de espacio	Permitir seleccionar el tipo de espacio (abierto o cerrado)
RF20	Visualización de rejilla	Permitir que se muestre u oculte la rejilla del espacio de evolución
RF21	Guardar simulación	Almacenar la configuración de la generación en curso del AC
RF22	Abrir archivo de simulación	Leer un archivo que contenga la configuración del AC
RF23	Color de células	Permitir la modificación del color de las células vivas, muertas y de proyección
RF24	Color de la rejilla	Permitir la modificación del color de la rejilla

Tabla 4.1: Requerimientos funcionales

4.2.4. Requerimientos no funcionales

Los requerimientos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema. Algunos de estos requerimientos pueden restringir el proceso que se utiliza para desarrollar el software [39], en este caso aplica para la utilización de un framework que permita desarrollar un solo simulador para los tres sistemas operativos (Windows, Linux y Mac OS). A continuación se muestran los requerimientos no funcionales.

Requerimiento	Nombre	Descripción
RNF01	Usabilidad	Proporcionar interfaces que permitan al usuario realizar las simulaciones con facilidad
RNF02	Multiplataforma	El sistema deberá poder ejecutarse en los sistemas Linux, Windows y Mac OS

Tabla 4.2: Requerimientos no funcionales

4.3. Herramientas utilizadas para el desarrollo

4.3.1. C++

El lenguaje programación C++ es una extensión del lenguaje C que tiene como propósito ayudar al manejo de objetos y fue creado por Bjarne Stroustrup y permite realizar un alto número de operaciones por segundo (1×10^8), por lo que es de gran utilidad para este sistema dada la necesidad de crear espacios de evolución de tamaños relativamente grandes y realizar la evaluación de las células.

4.3.2. Qt Creator

Es un IDE (entorno de desarrollo integrado) multiplataforma que permite a los desarrolladores crear aplicaciones para escritorio, móviles y plataformas integradas, y por tanto permite un mejor manejo de instrucciones para cada sistema operativo (Linux, MacOS y Windows) que además simplifica la creación de la interfaz de usuario.

4.4. Arquitectura

Según los requerimientos obtenidos y propuestos, se ha determinado separar la funcionalidad e interfaz de usuario en 5 clases:

- MainWindow: Tiene como propósito mostrar los controles para establecer las configuraciones iniciales, así como mostrar las gráficas con los puntos obtenidos para cada una.
- MatrixWindow: Su objetivo es contener un widget del tipo *EvolutionSpace* y manejar los eventos para esa ventana.
- WidgetEvolutionSpace: Su funcionalidad radica en mostrar el espacio de evoluciones con las células vivas, muertas y de evolución, así como la rejilla.
- CellularAutomata: Su propósito es realizar las evaluaciones de cada célula en el espacio de evolución de acuerdo con la configuración inicial, tipo de vecindad y tipo de espacio.
- Polynomial: Su objetivo es calcular el polinomio característico mediante la teoría del campo promedio.

4.4.1. Diagrama de clases

A continuación en Fig. 4.4.1se presenta el diagrama de clases que tiene como objetivo mostrar las relaciones entre las mismas, así como los atributos y métodos de cada una.

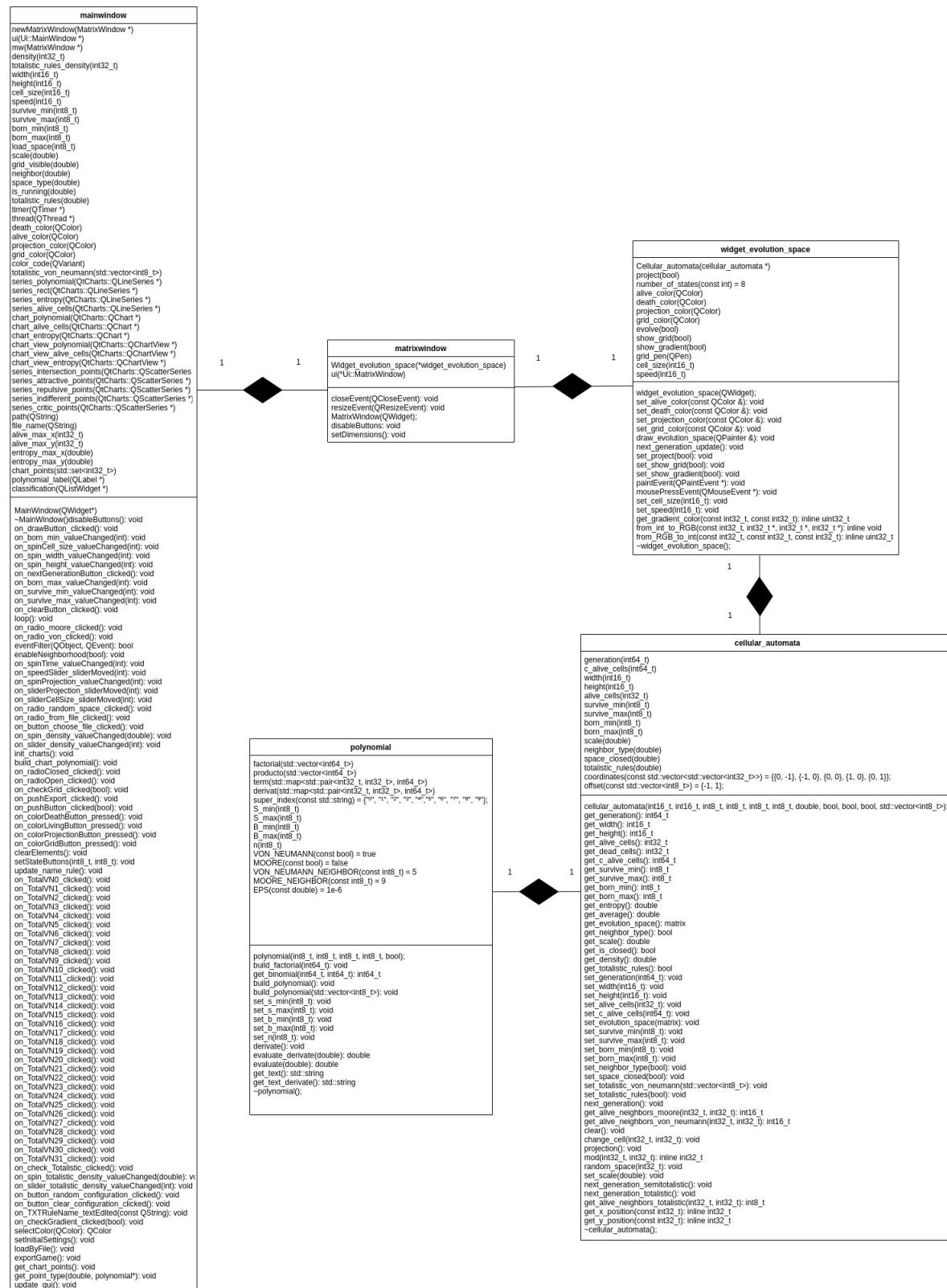


Fig. 4.4.1: Diagrama de clases completo

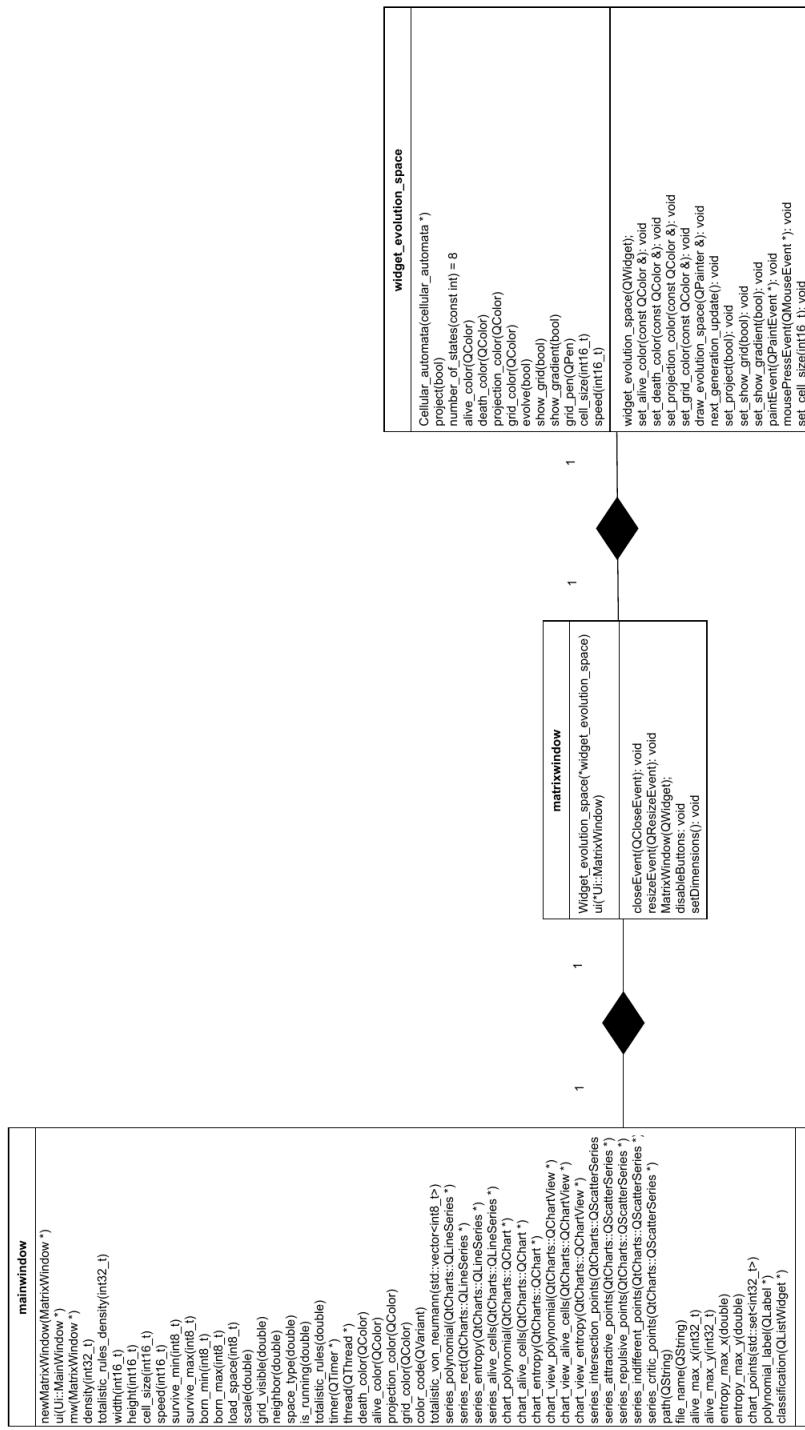


Fig. 4.4.2: Diagrama de clases

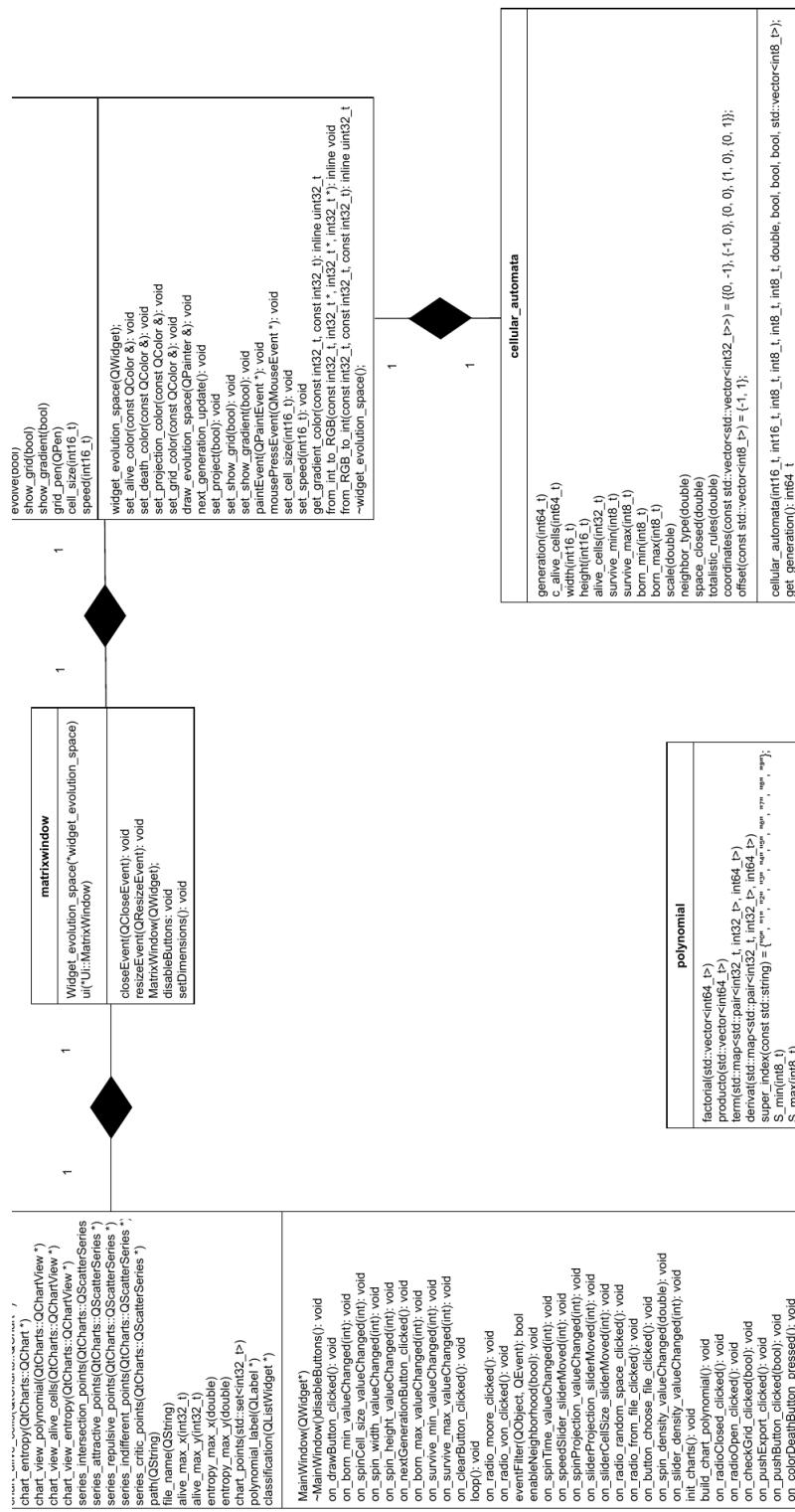


Fig. 4.4.3: Diagrama de clases (continuación)

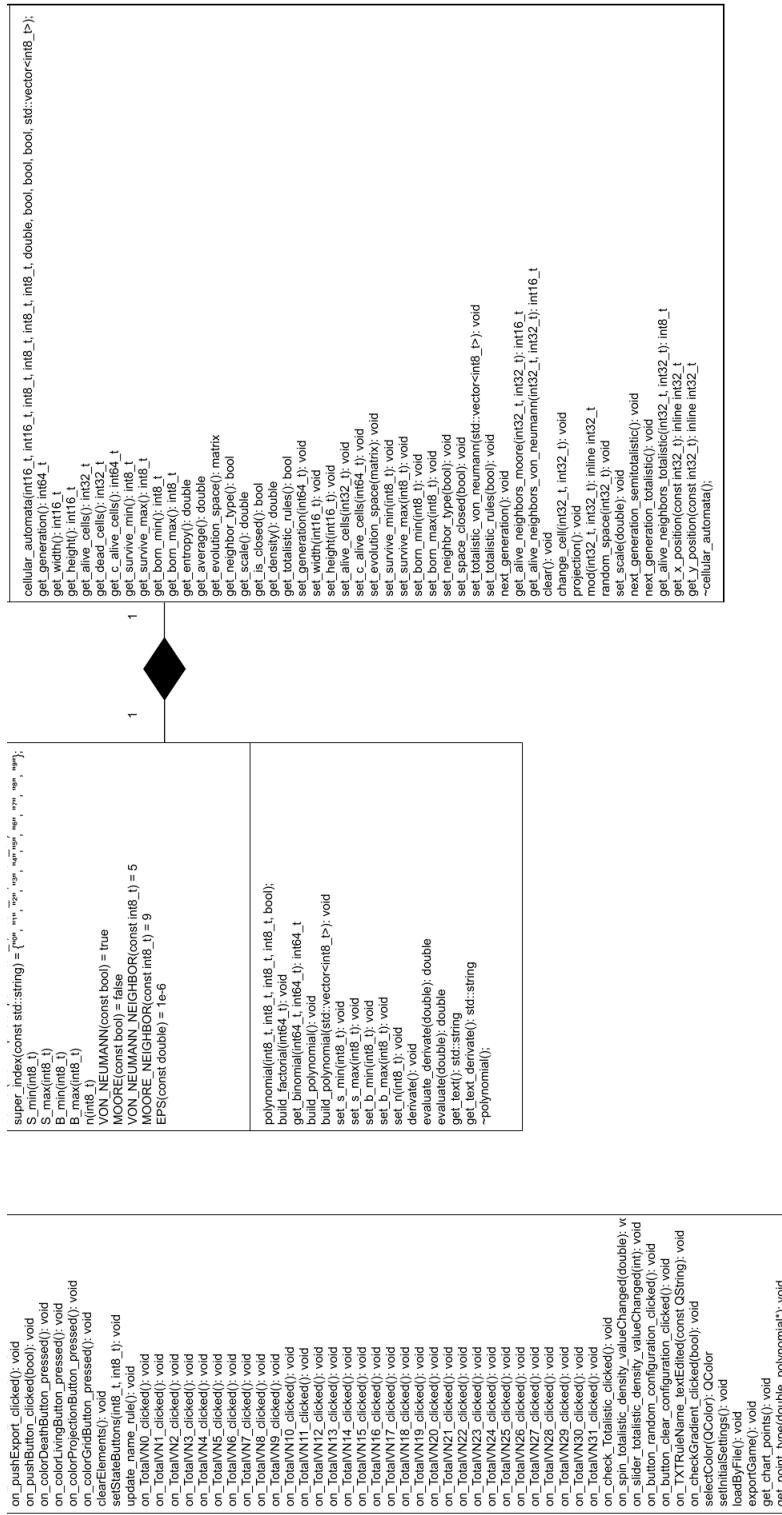


Fig. 4.4.4: Diagrama de clases (continuación)

4.4.2. Diagrama BPMN

Antes de pasar a la fase de prototipado, se hará una breve presentación del diagrama BPMN diseñado para entender cómo funcionará el simulador y las interacciones que se existen para que tenga un correcto funcionamiento.

BPMN es un acrónimo en inglés (Business Process Model and Notation) que significa Modelo y Notación de Procesos de Negocio, el cual permite precisamente describir el proceso de negocio generalmente utilizando un diagrama que muestre el flujo del trabajo, para los cuales se describen algunas tareas y subprocessos que implican para que el sistema funcione de forma adecuada y nos permita ver las dimensiones del sistema, así como planificar de una forma más eficiente el desarrollo de cada módulo.

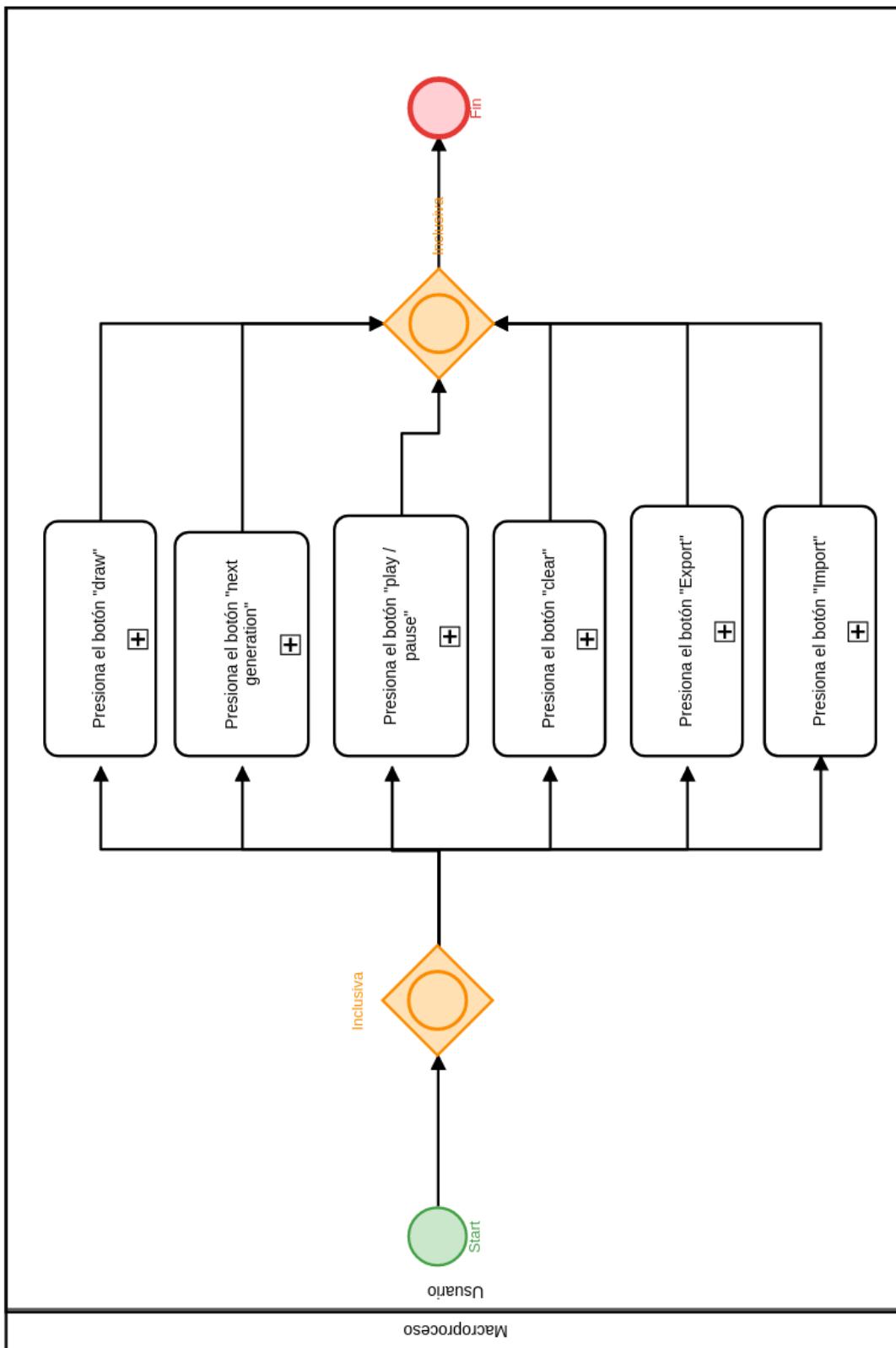


Fig. 4.4.5: Macroprocreso

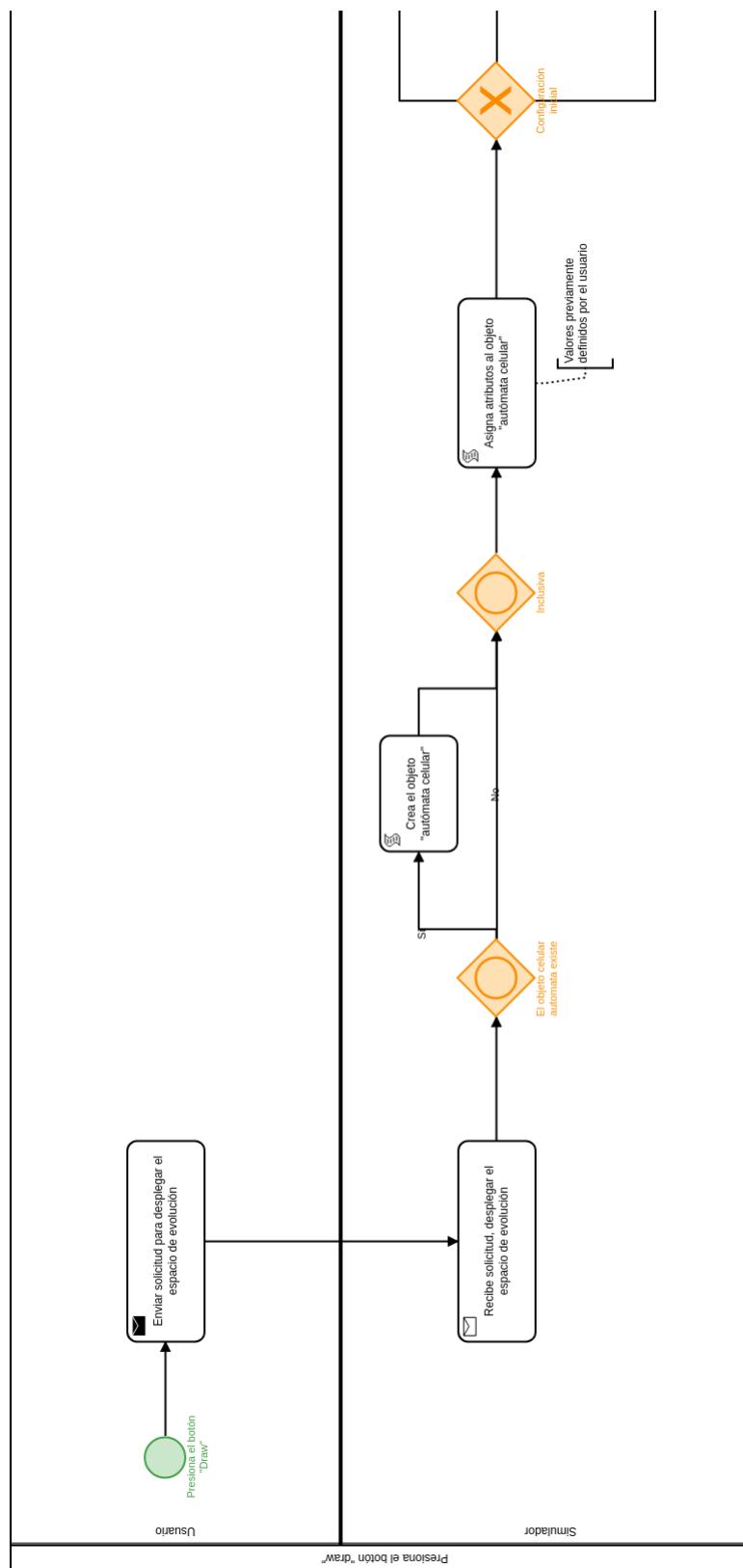


Fig. 4.4.6: Subproceso - Draw

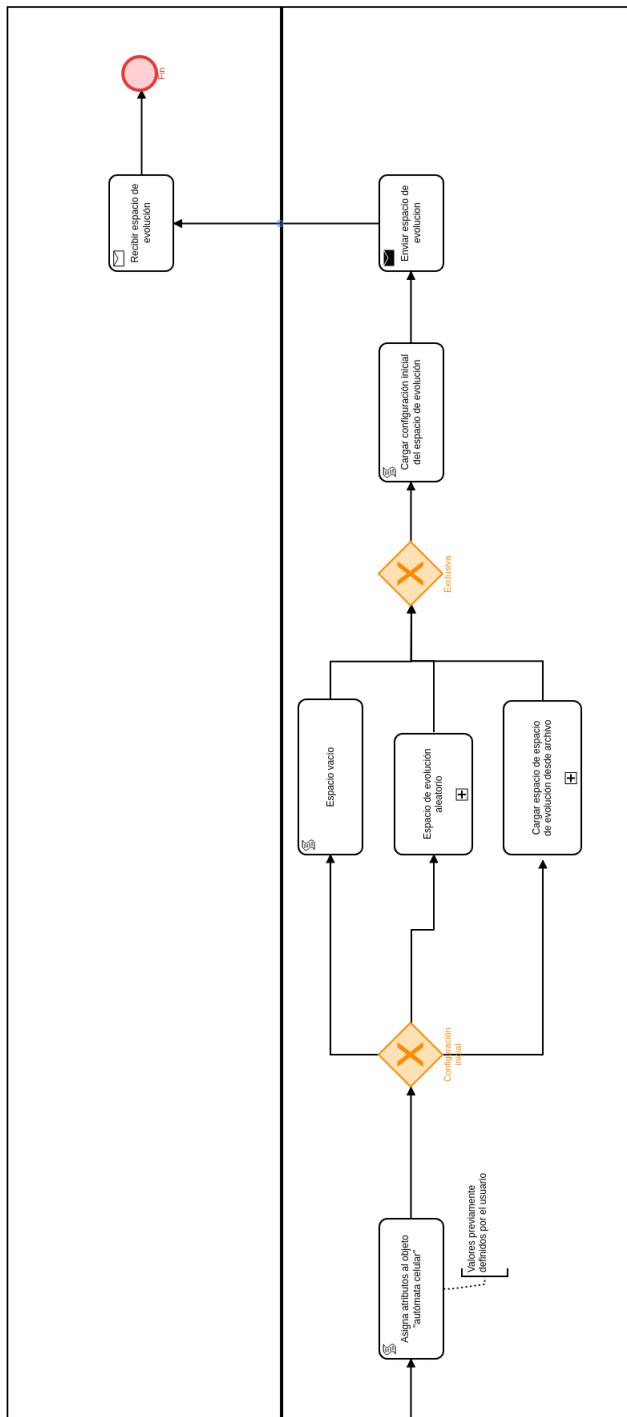


Fig. 4.4.7: Subproceso - Draw (continuación)

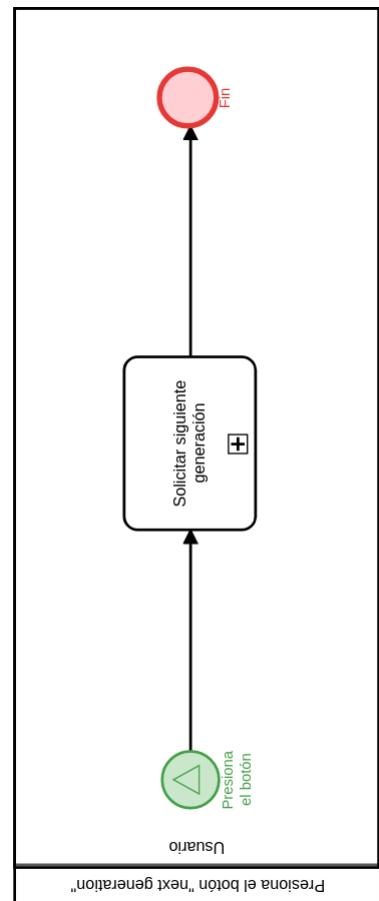


Fig. 4.4.8: Subproceso - Next generation

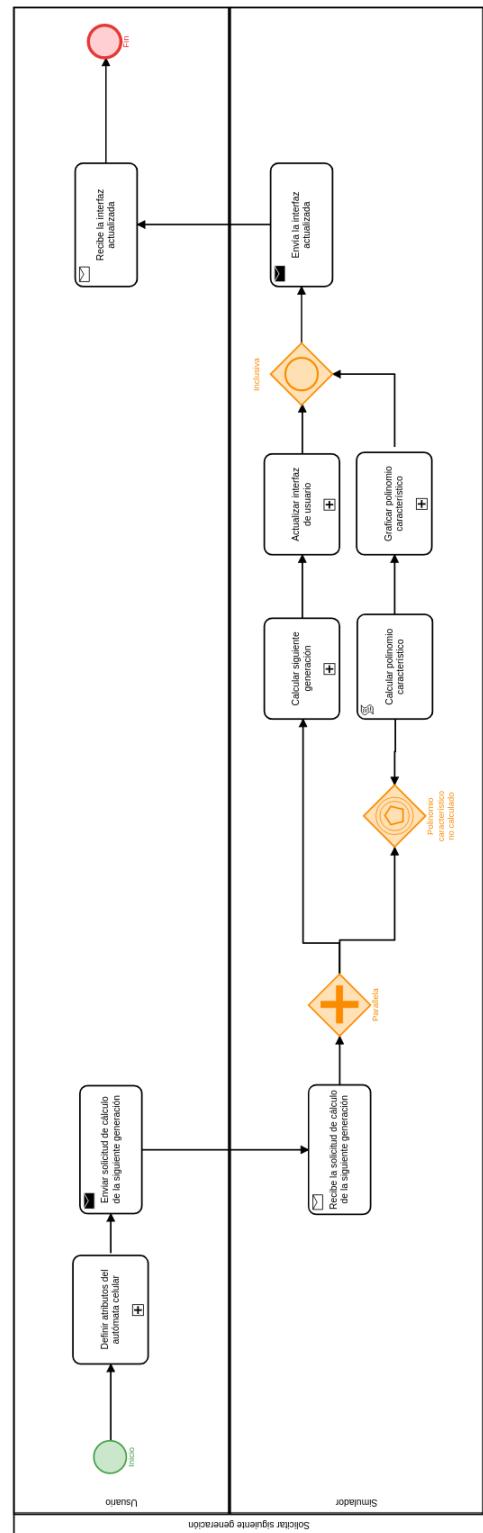


Fig. 4.4.9: Subproceso - Solicitar siguiente generación

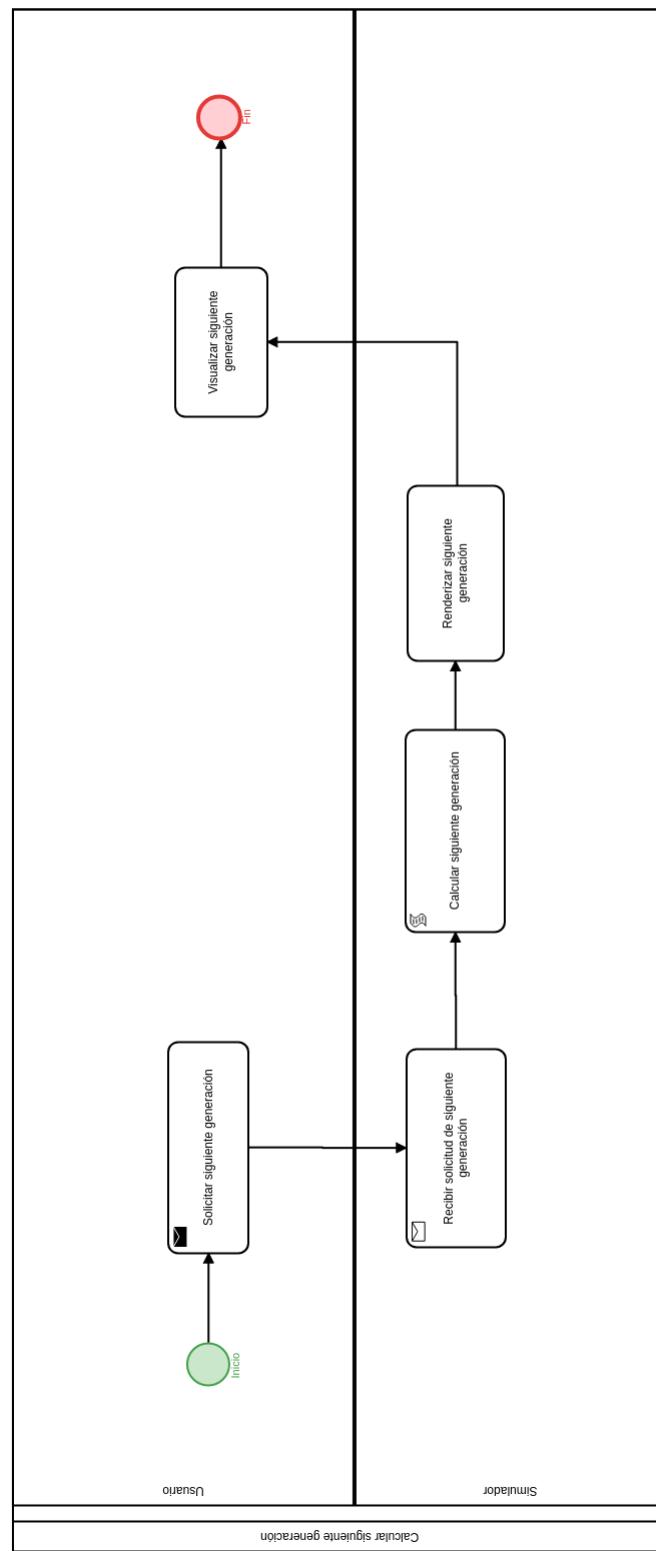


Fig. 4.4.10: Subproceso - Calcular siguiente generación

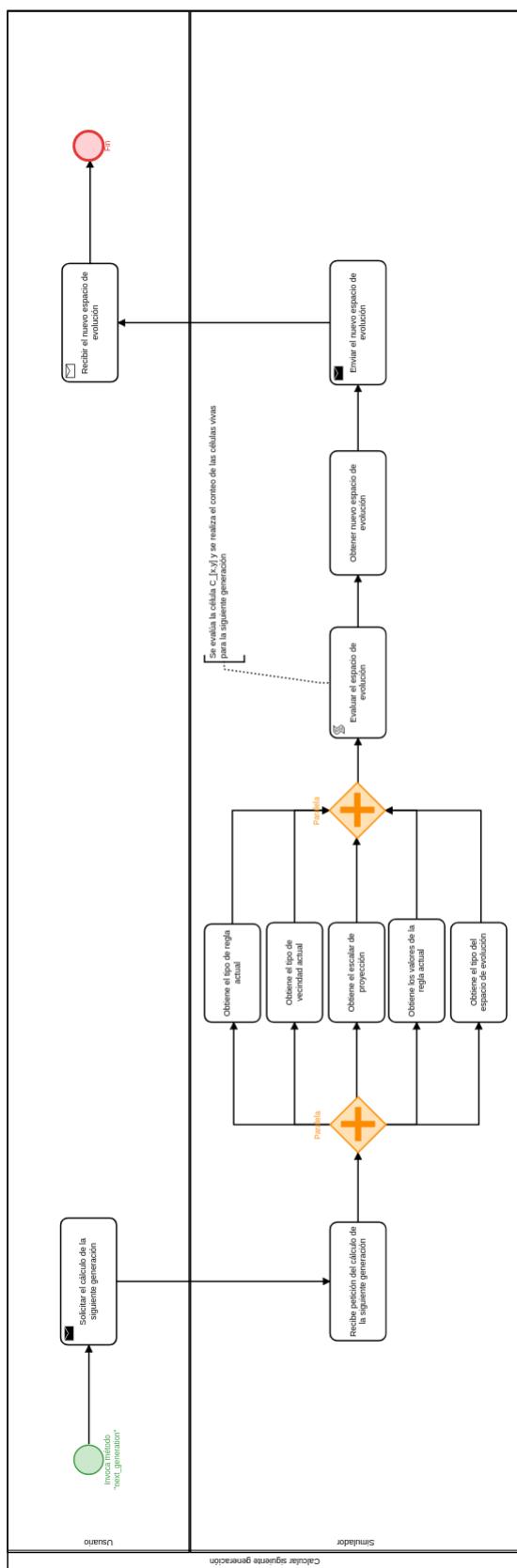


Fig. 4.4.11: Subproceso - Calcular siguiente generación (continuación)

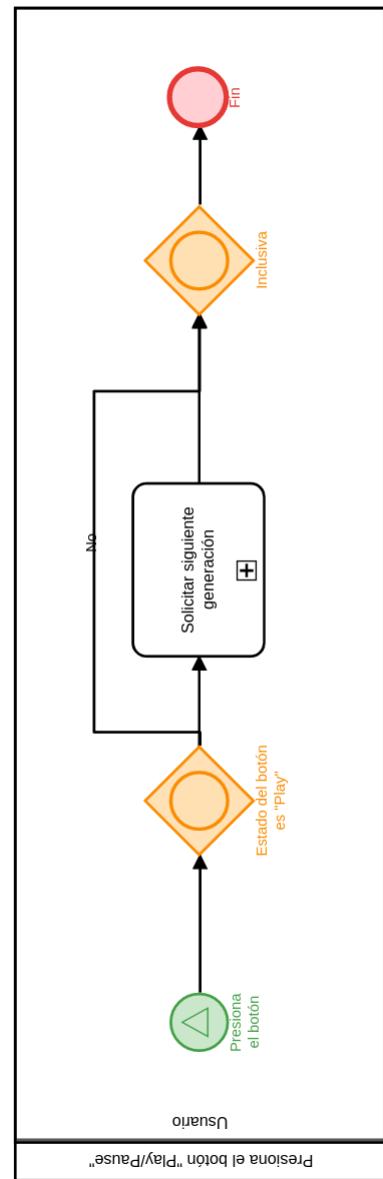


Fig. 4.4.12: Subproceso - Play/pause

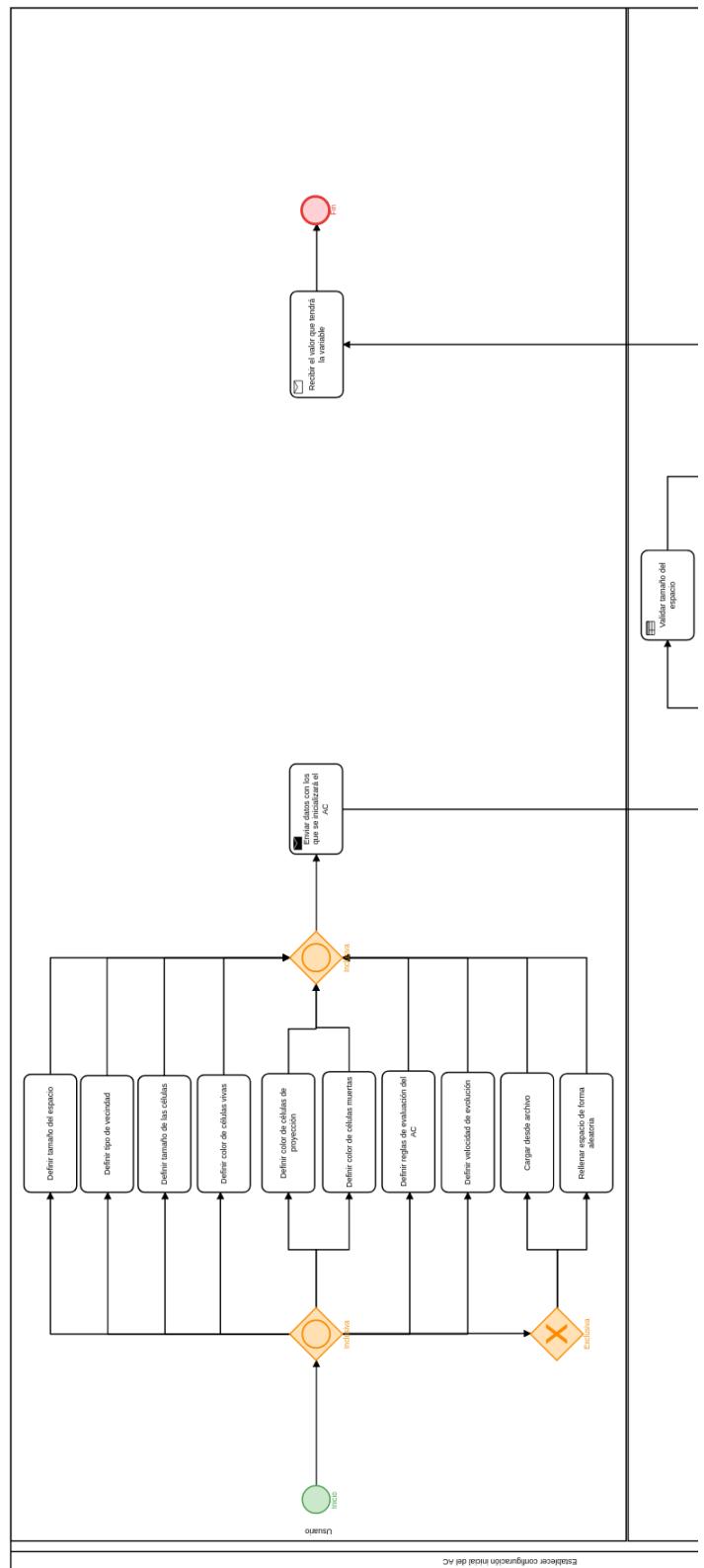


Fig. 4.4.13: Subproceso - Establecer configuración inicial

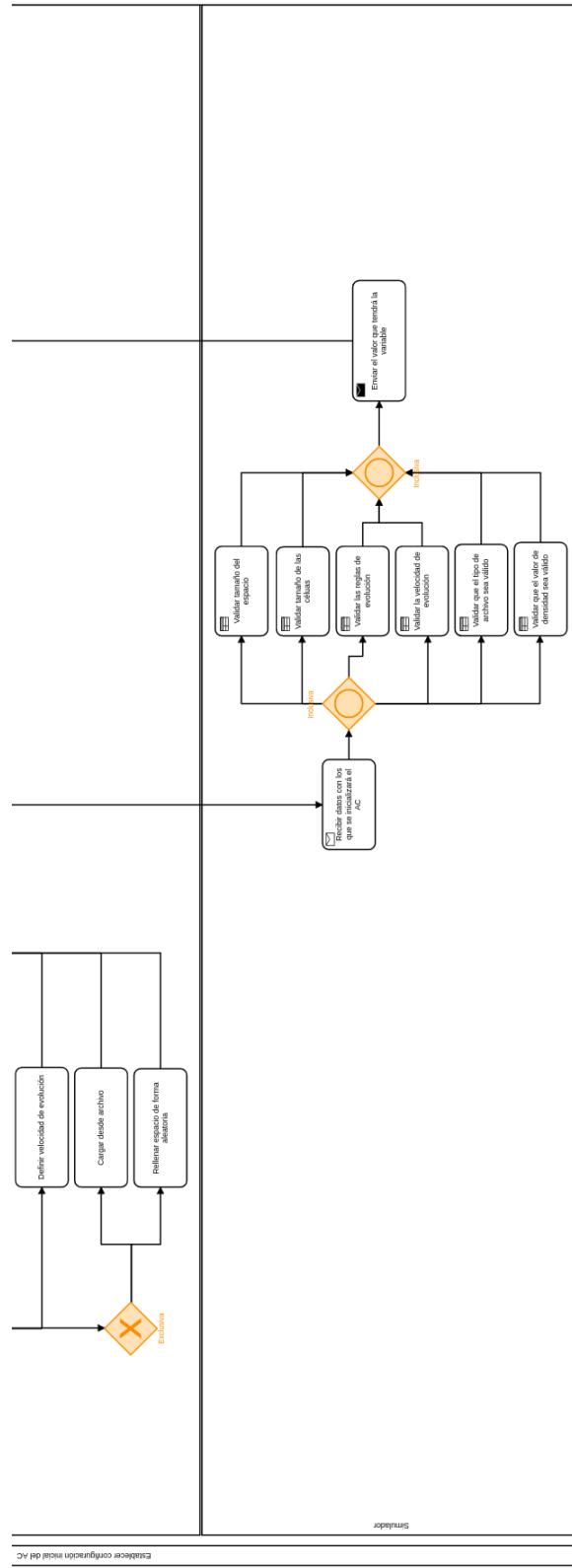


Fig. 4.4.14: Subproceso - Establecer configuración inicial (continuación)

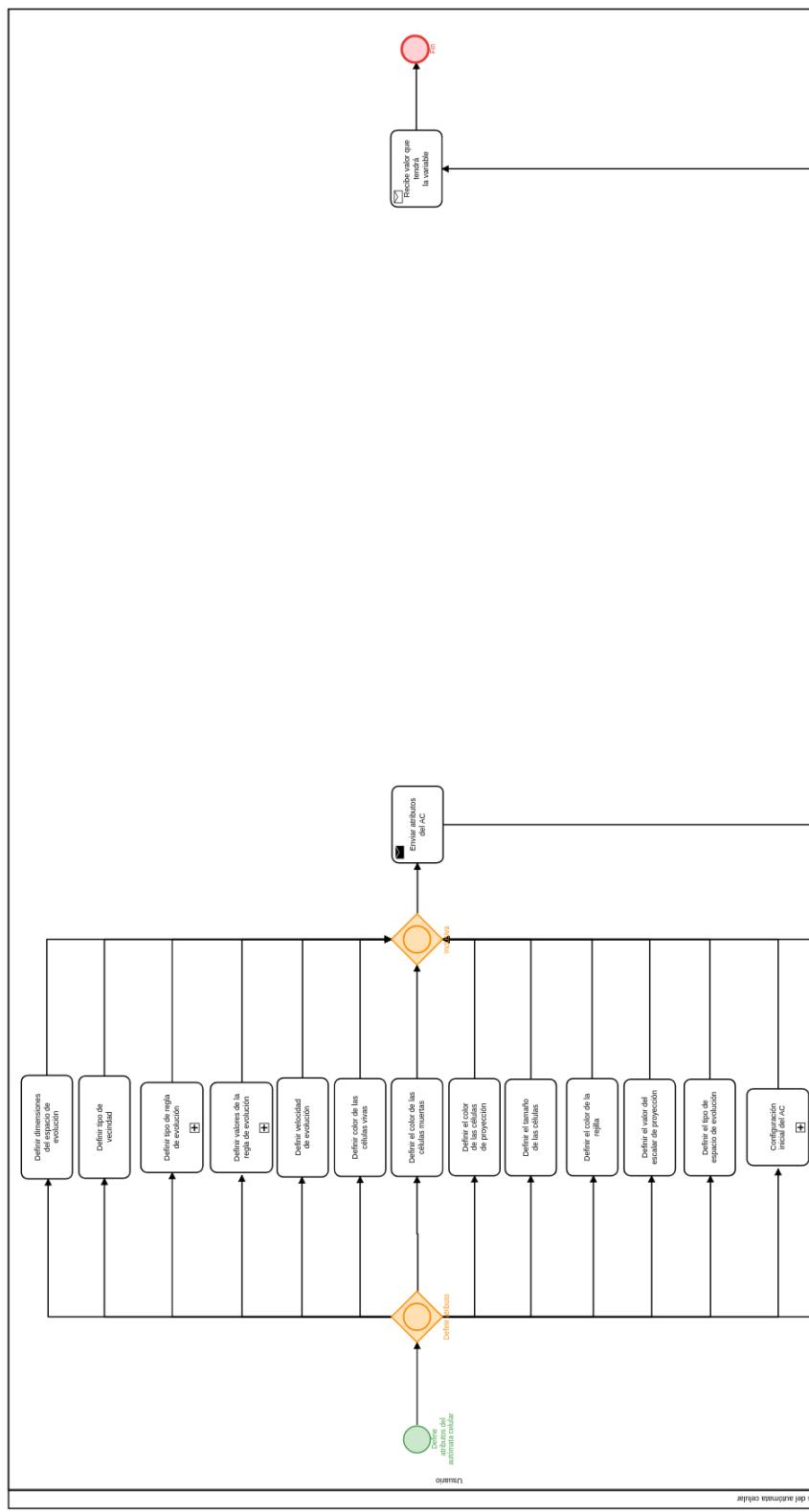


Fig. 4.4.15: Subproceso - Definir atributos del autómata celular

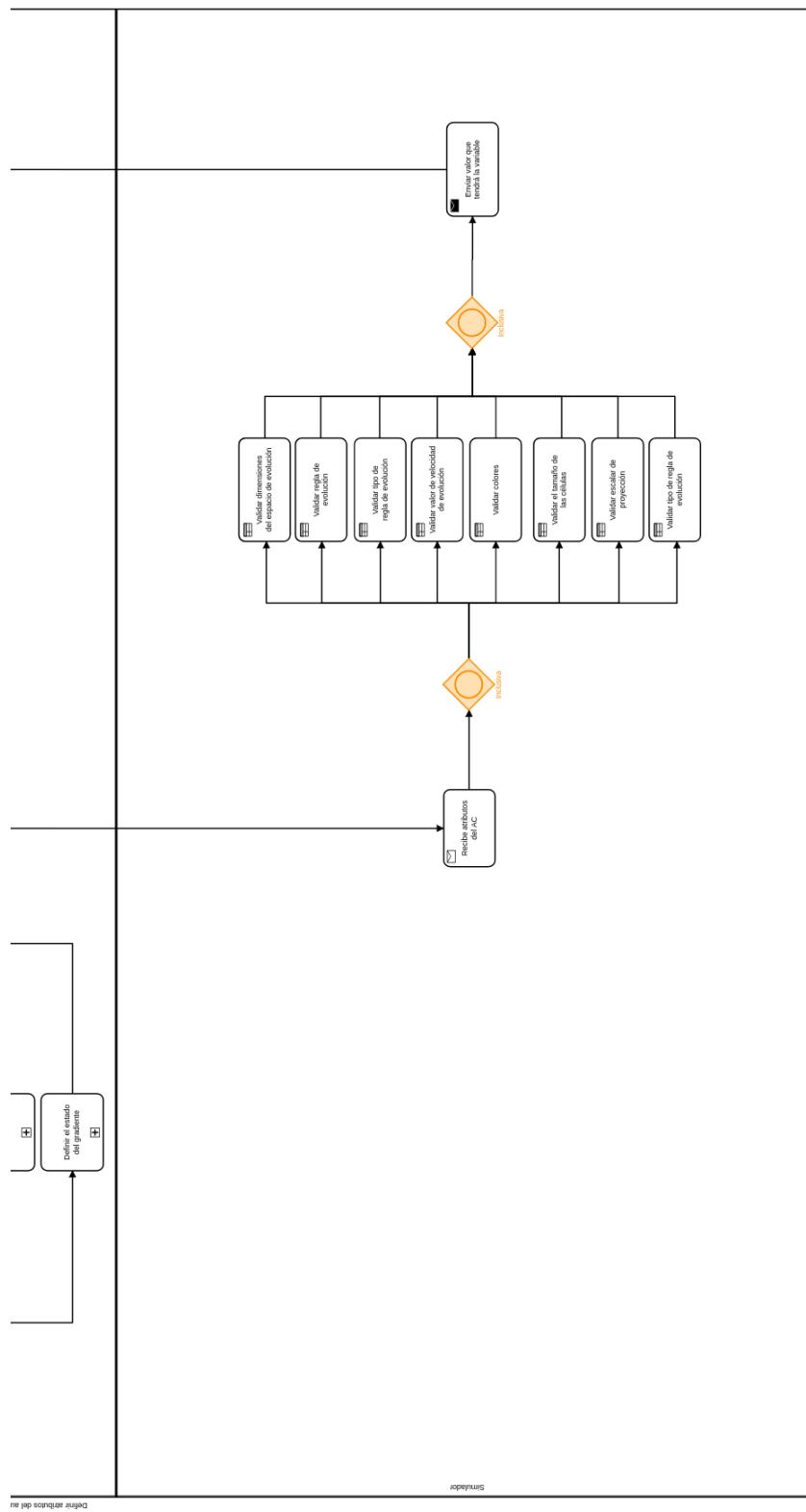


Fig. 4.4.16: Subproceso - Definir atributos del autómata celular (continuación)

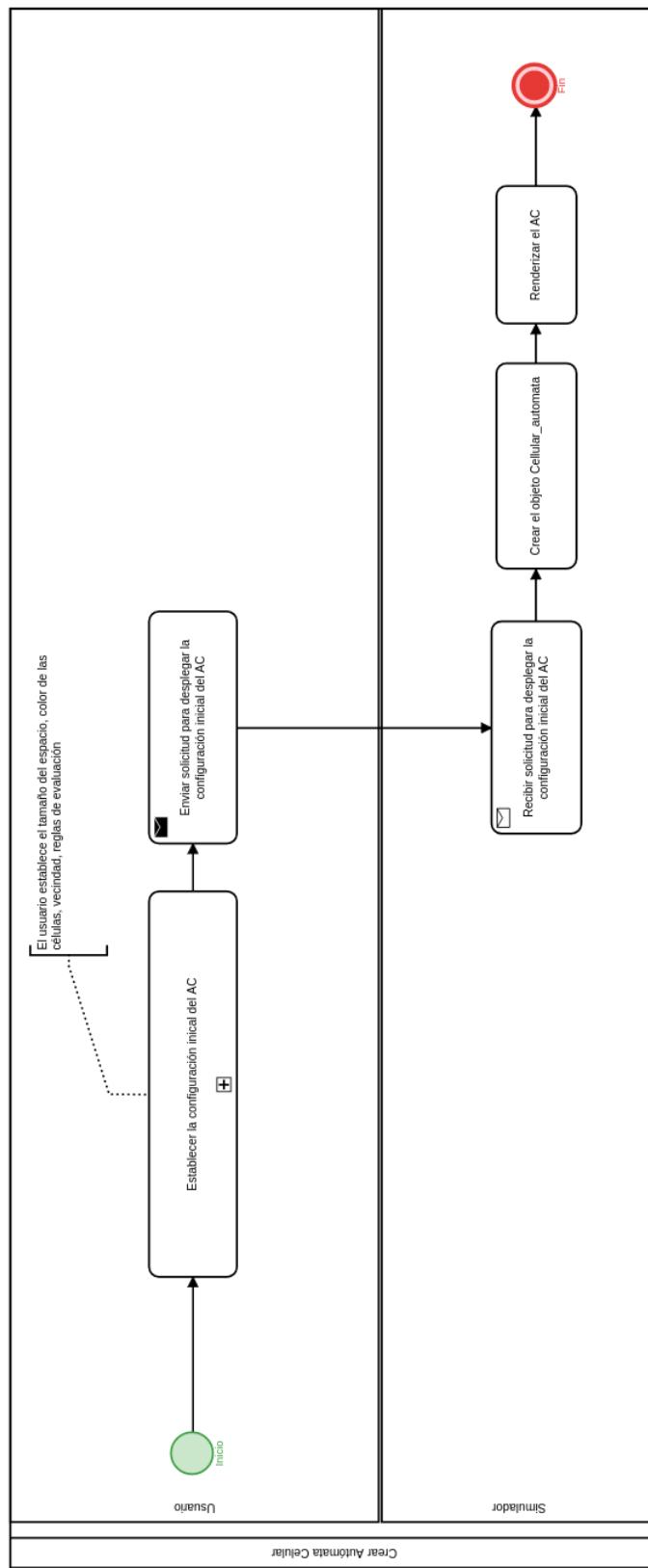


Fig. 4.4.17: Subproceso - Crear autómata celular

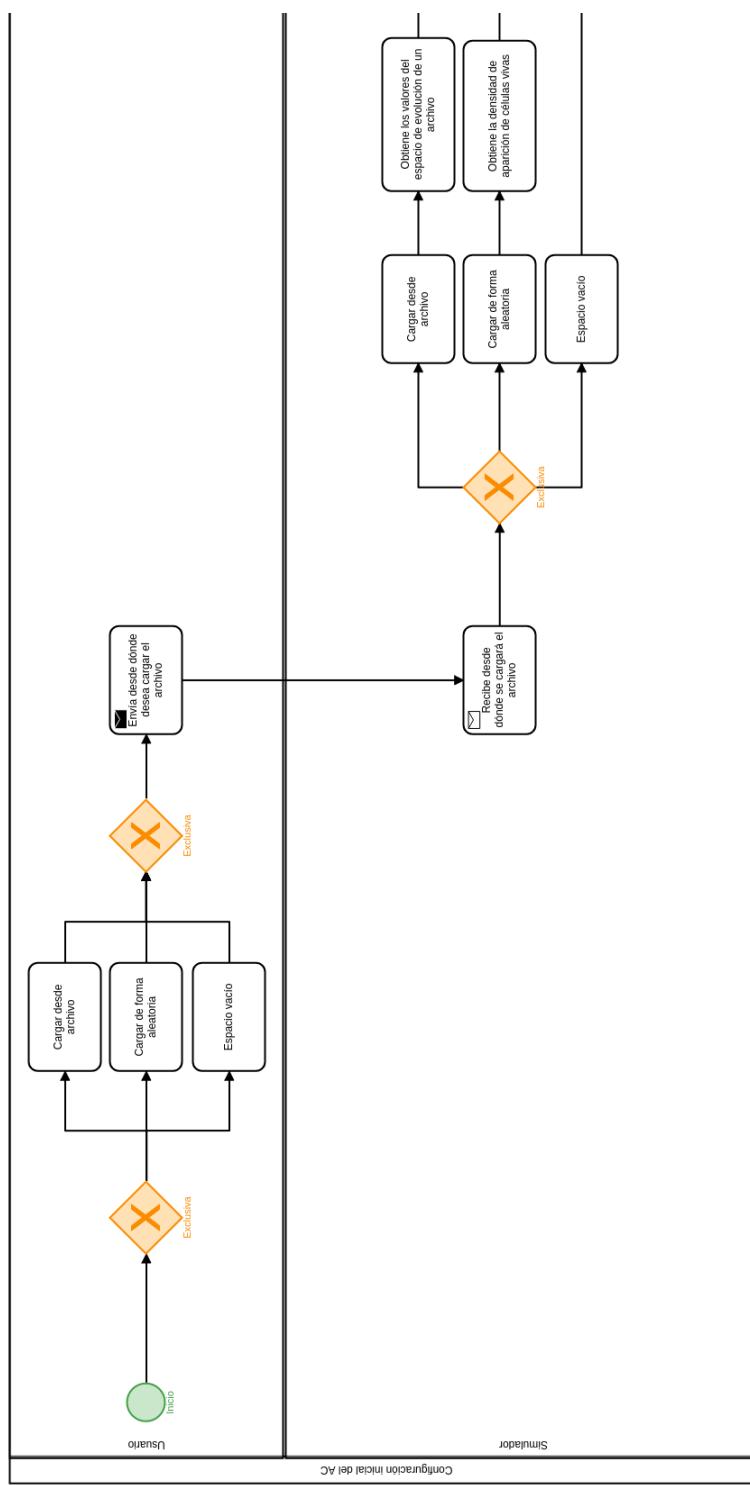


Fig. 4.4.18: Subproceso - Configuración inicial del autómata celular

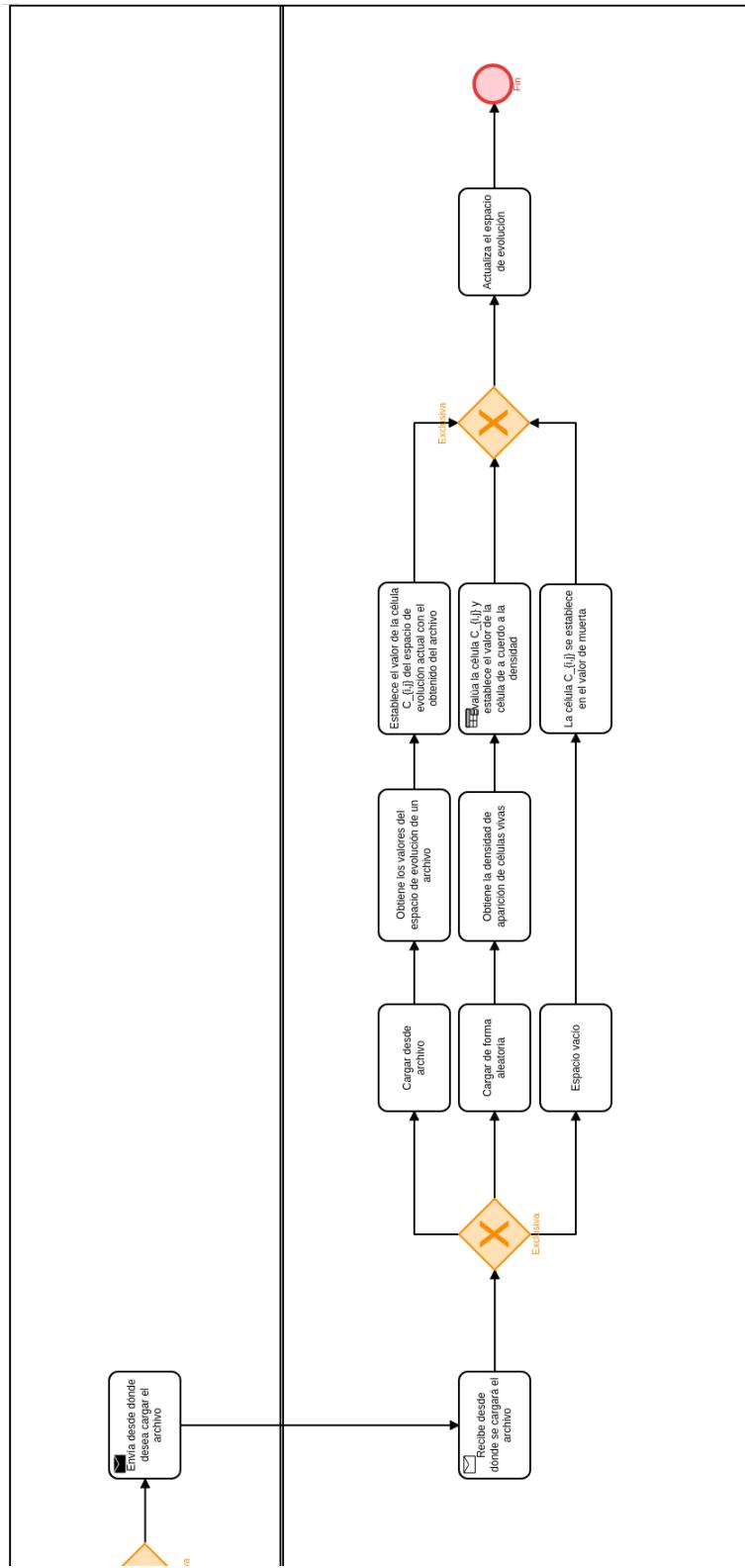


Fig. 4.4.19: Subproceso - Configuración inicial del autómata celular (continuación)

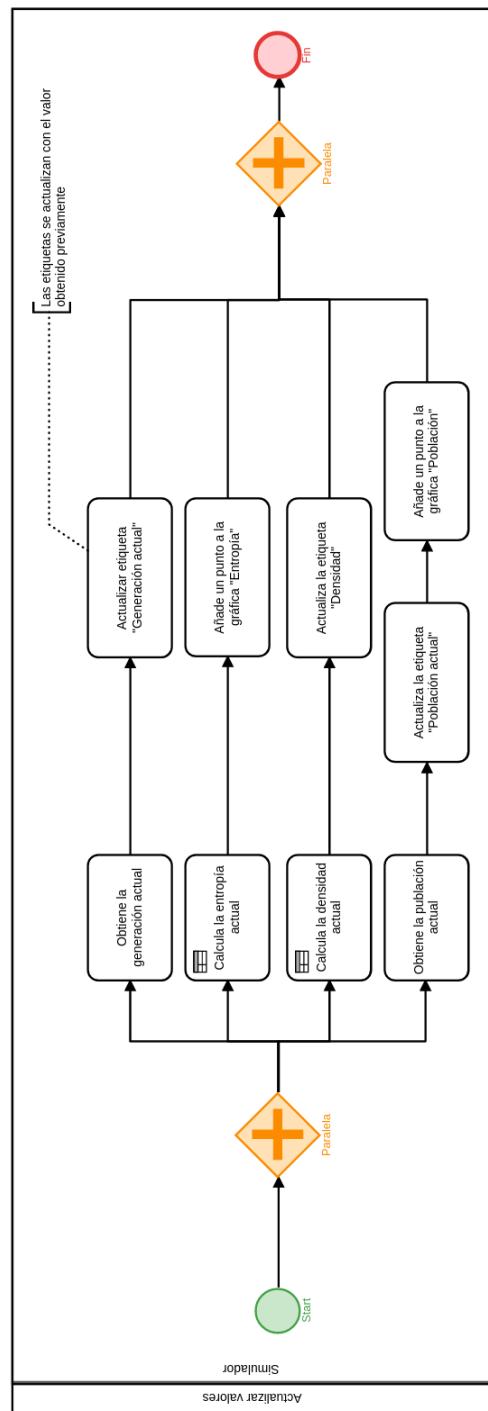


Fig. 4.4.20: Subproceso - Actualizar valores

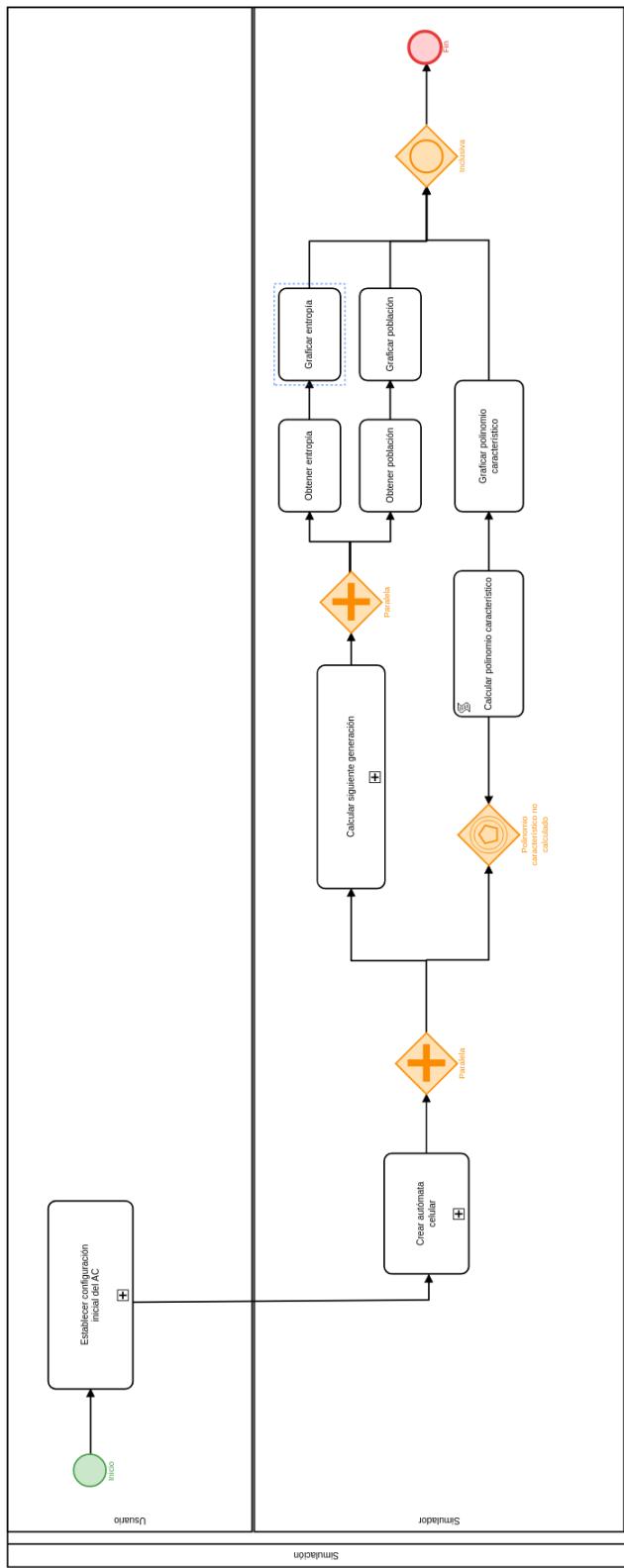


Fig. 4.4.21: Subproceso - Simulación

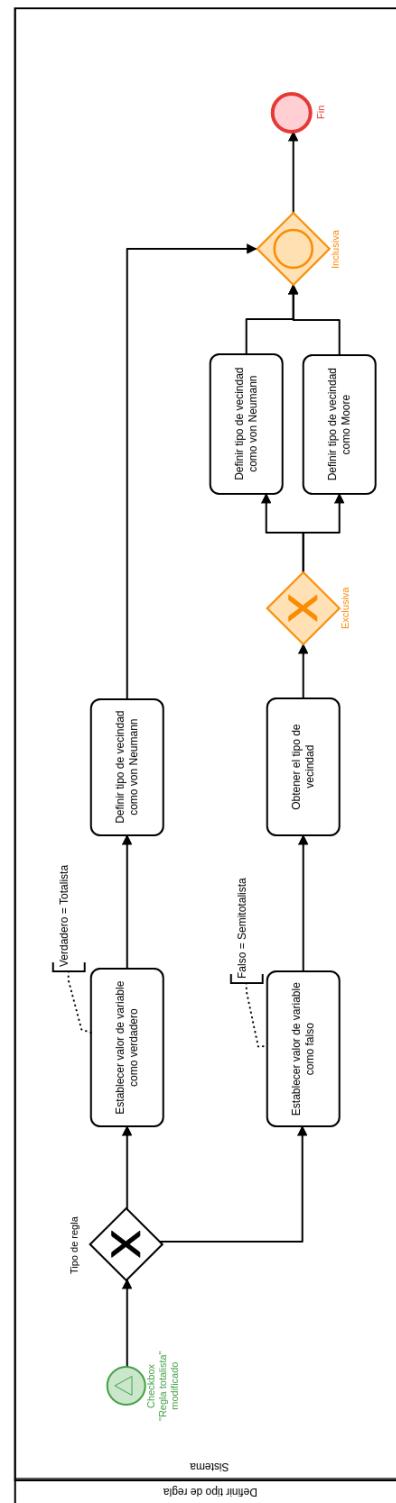


Fig. 4.4.22: Subproceso - Definir tipo de regla

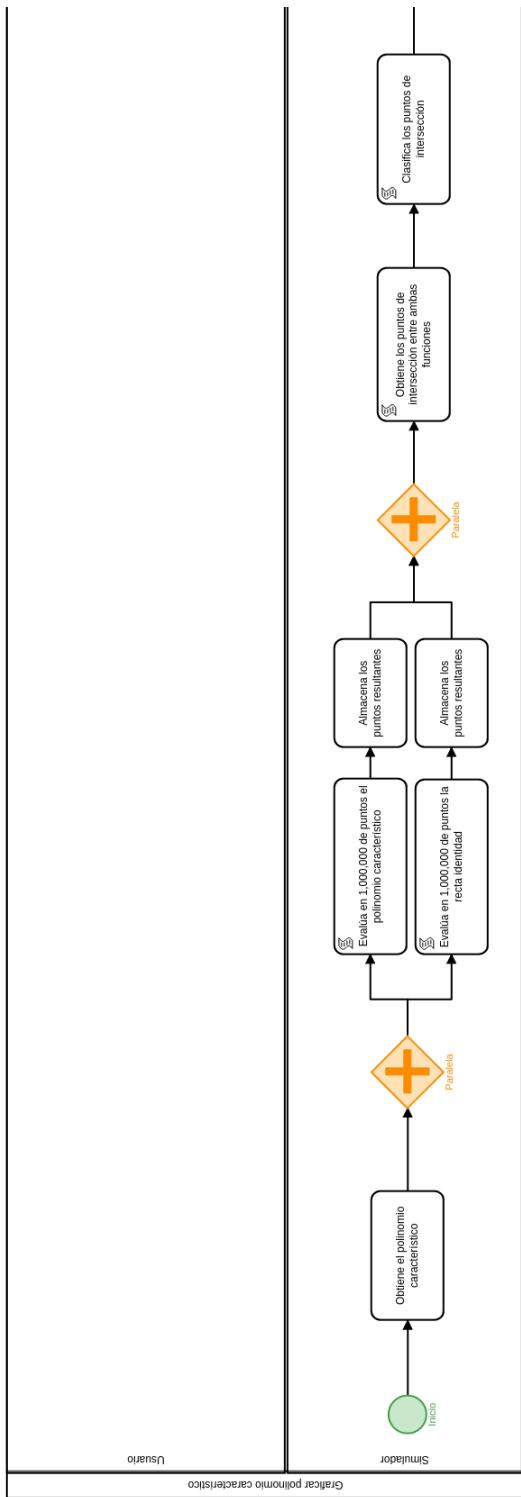


Fig. 4.4.23: Subproceso - Graficar polinomio característico

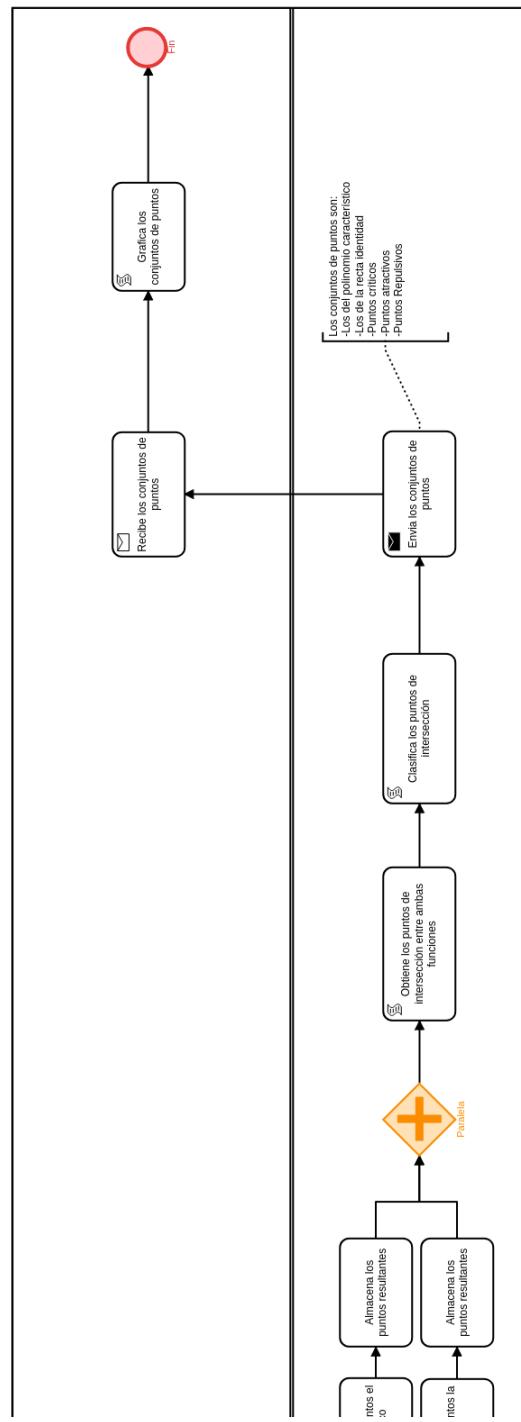


Fig. 4.4.24: Subproceso - Graficar polinomio característico (continuación)

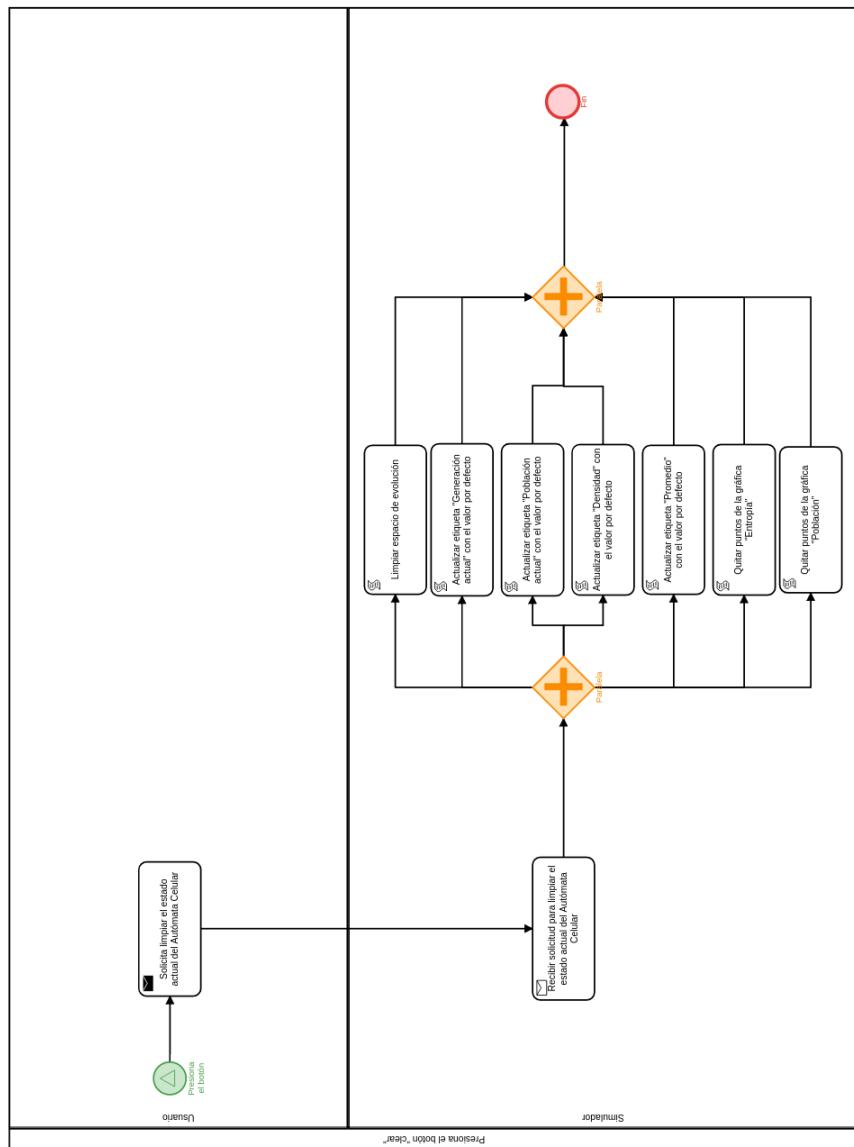


Fig. 4.4.25: Subproceso - Clear

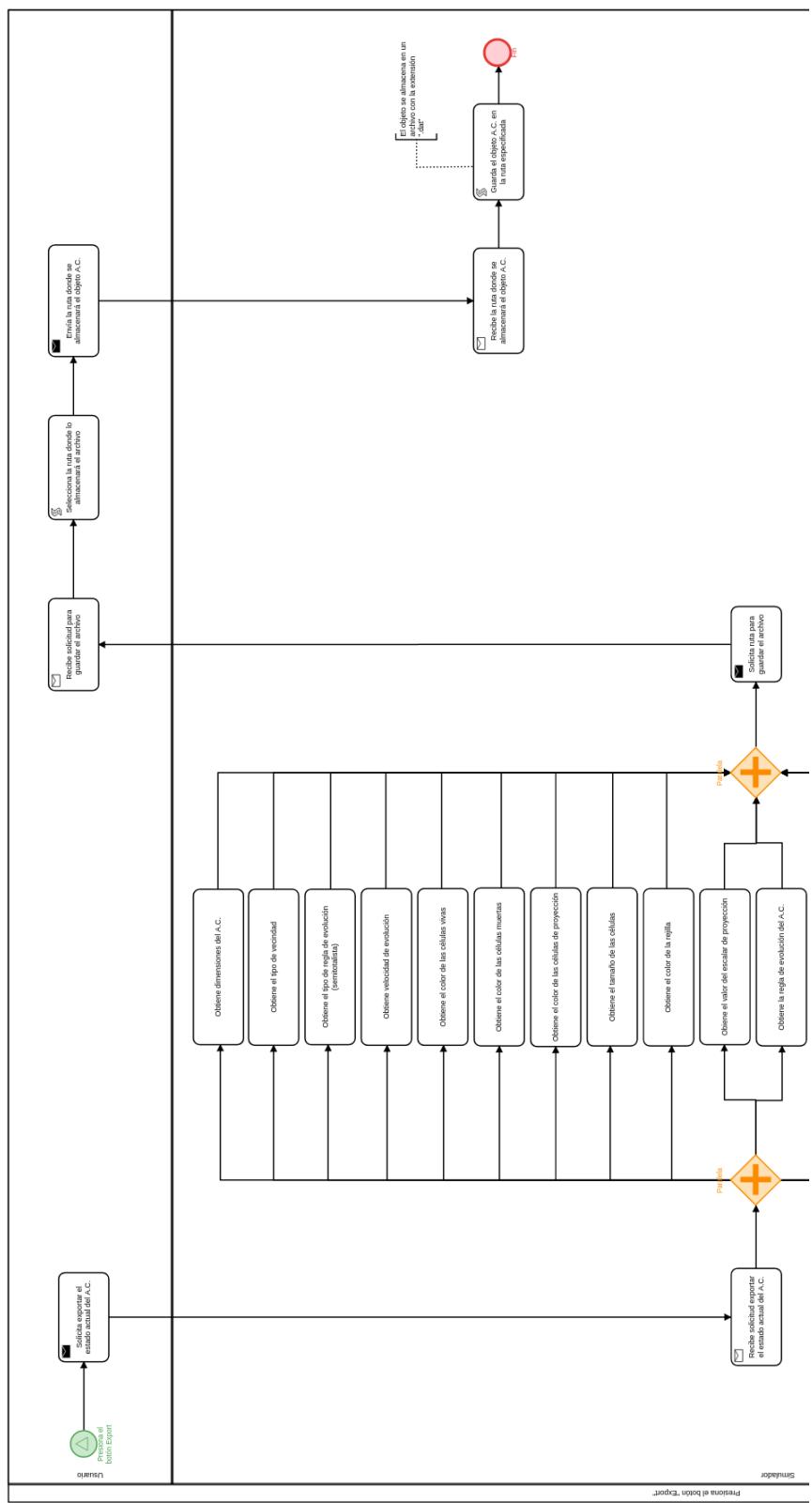


Fig. 4.4.26: Subproceso - Export

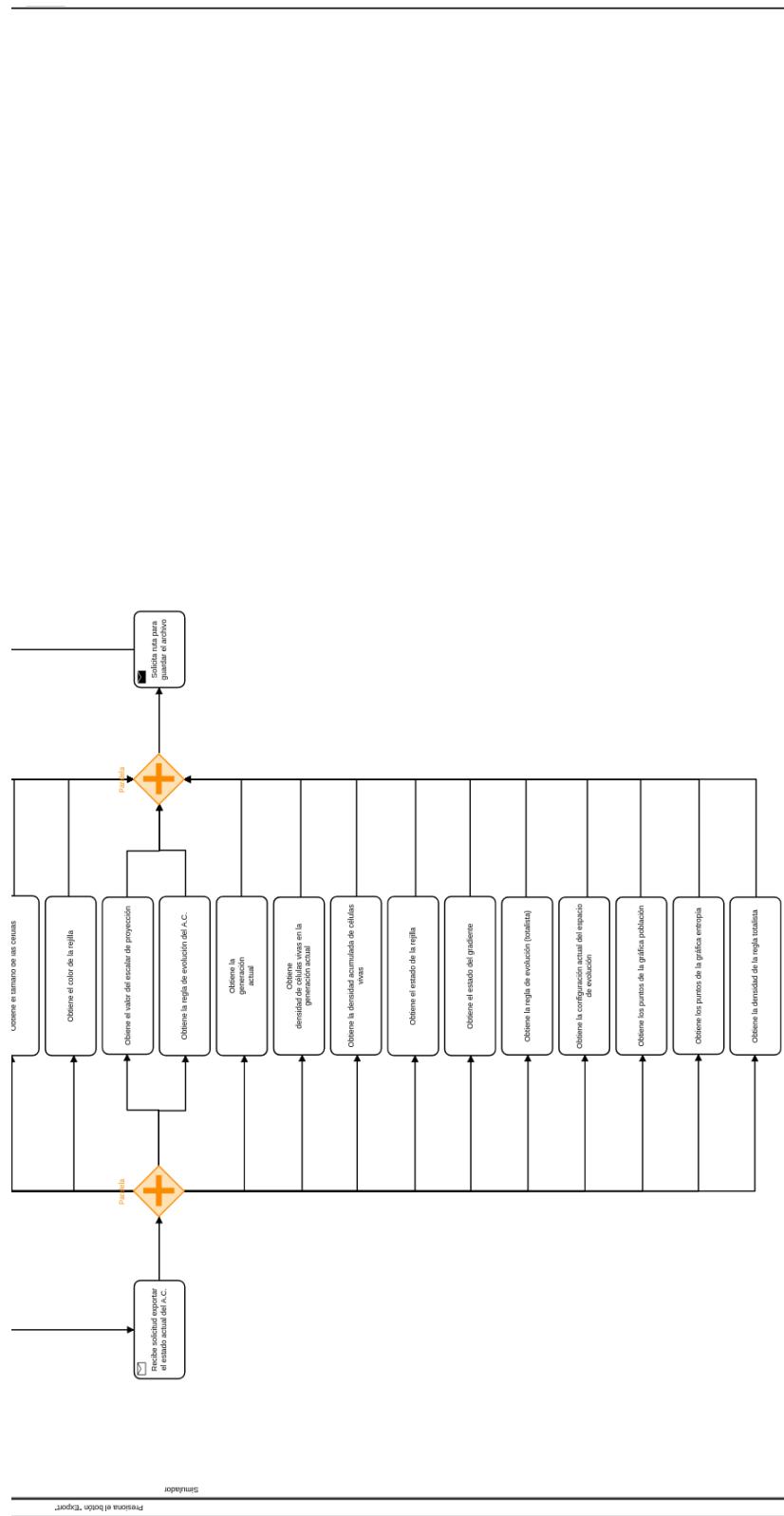


Fig. 4.4.27: Subproceso - Export (continuación)

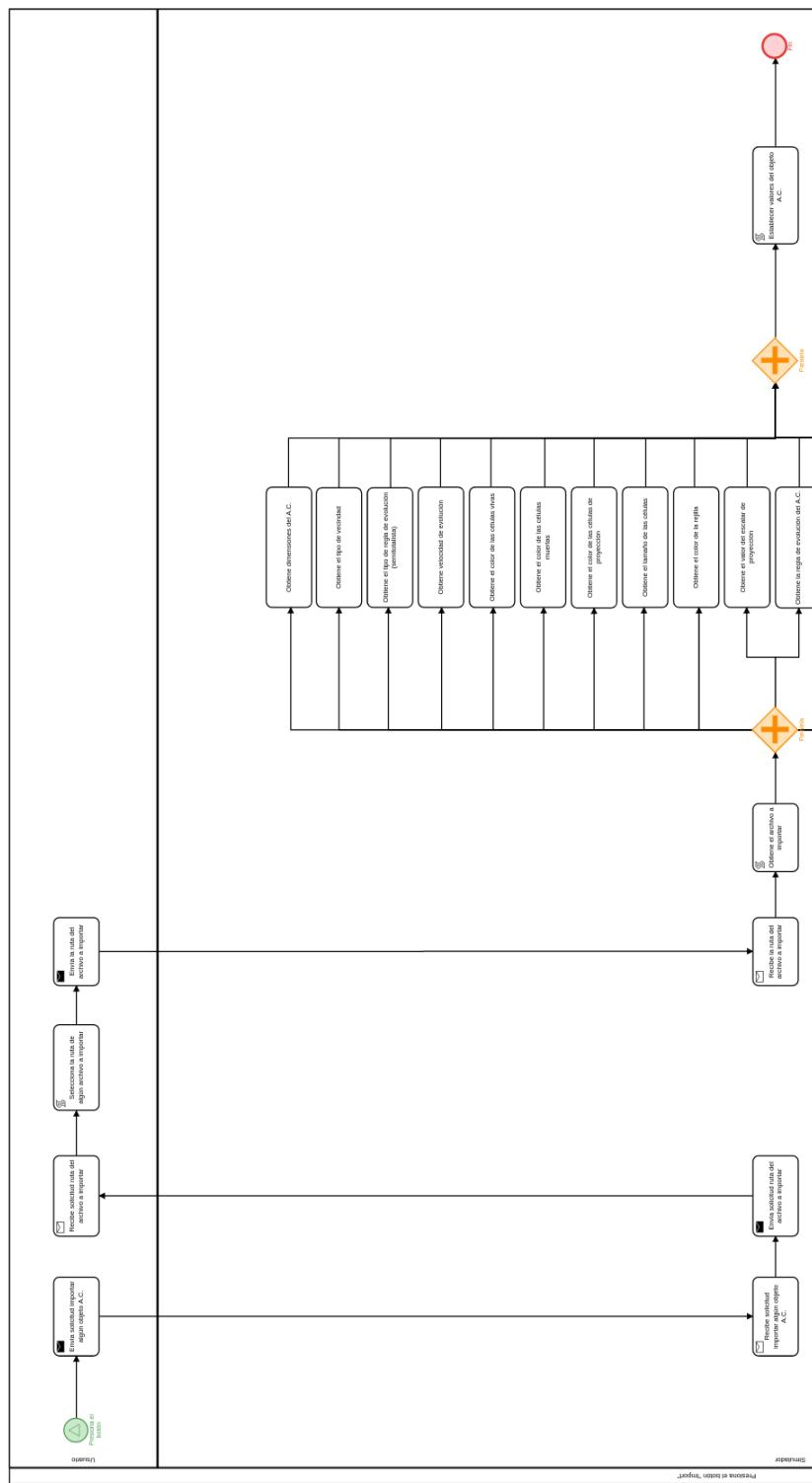


Fig. 4.4.28: Subproceso - Import

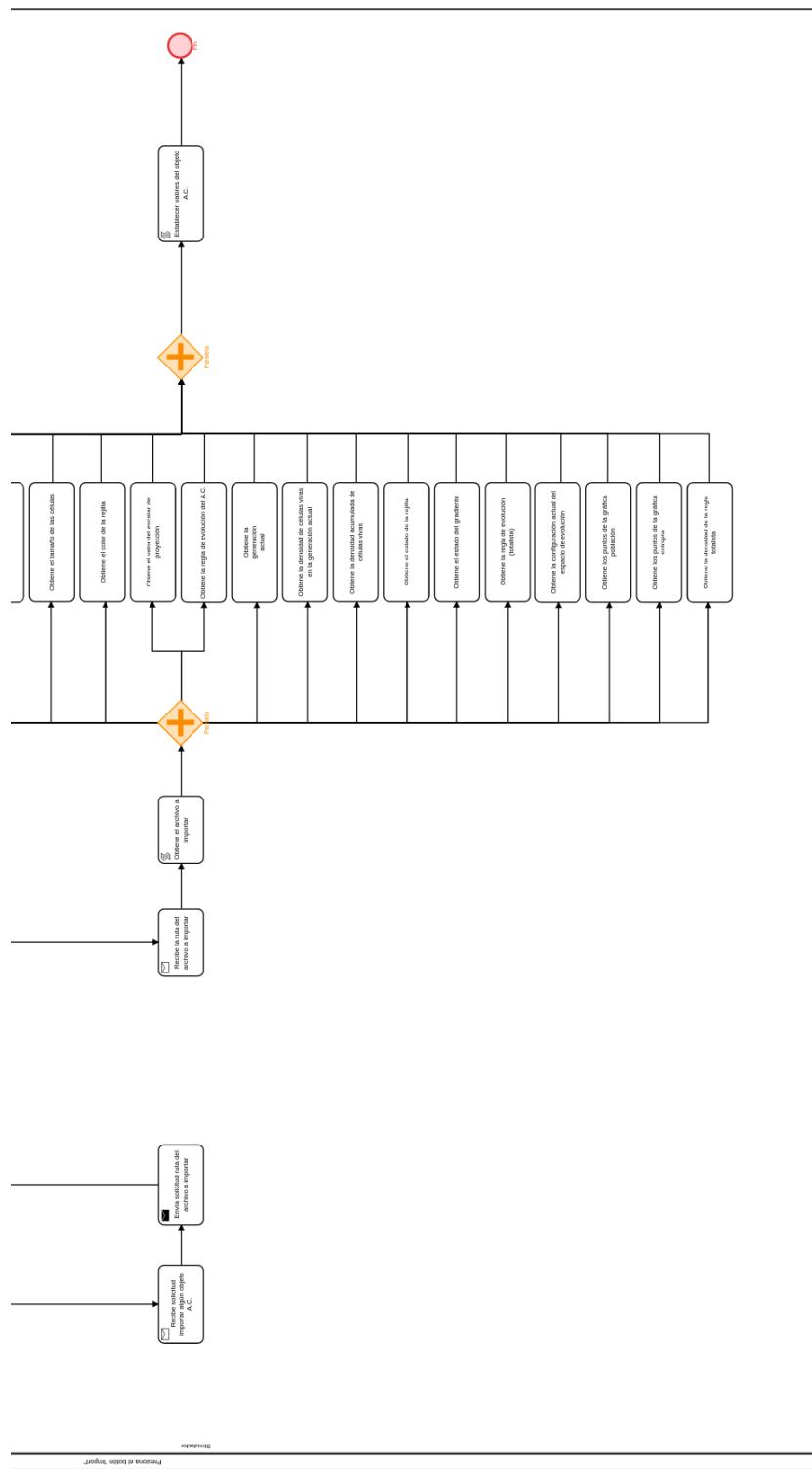


Fig. 4.4.29: Subproceso - Import (continuación)

4.5. Casos de uso

A continuación se presenta una descripción detallada de los casos de uso que conforman el sistema y la interacción de cada uno de los elementos que lo componen de acuerdo a lo modelado en el diagrama BPMN, utilizando además reglas de negocio para que los desarrolladores puedan comprender de una forma más exacta lo que se quiere lograr con el sistema.

4.5.1. CU01 Definir tamaño del espacio

Descripción completa

Un AC bidimensional trabaja sobre una cuadrícula con un número definido de células $N \times M$. El usuario define los valores de N y M , es decir, la altura y anchura de la matriz que representa el espacio.

Atributos importantes

Caso de Uso:	CU01 Definir tamaño del espacio
Versión:	1.2
Actor:	Usuario
Propósito:	Definir el tamaño del espacio del AC
Resumen:	El usuario indica el tamaño de la matriz que utilizará el AC (altura y anchura)
Entradas:	Altura, anchura
Salidas:	Dimensión del espacio del AC
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Esquivel Valdez Alberto
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 Ingresa el valor de la altura vía IU-CU01 Definir altura del espacio con base en RN01 Valor permitido para la dimensión del espacio [Trayectoria Alternativa CU01.1]
- 2 El sistema valida que el valor ingresado sea correcto con base en RN01 Valor permitido para la dimensión del espacio [Trayectoria Alternativa CU01.2]
- 3 El sistema actualiza el valor de la variable que almacena la altura con el ingresado por el usuario
- - - *Fin del caso de uso.*

Trayectoria alternativa CU01.1:

Condición: Ingresar el valor de la anchura

- CU01.1 - 1** Ingresa el valor de la anchura vía IU-CU01.1 Definir anchura del espacio
- CU01.1 - 2** El sistema valida que el valor ingresado sea correcto con base en RN01 Valor de dimensión del espacio [Trayectoria Alternativa CU01.3]
- CU01.1 - 3** El sistema actualiza el valor de la variable que almacena la anchura con el ingresado por el usuario
- - - *Fin de la trayectoria.*

Trayectoria alternativa CU01.2:

Condición: Ingresó un valor incorrecto para la altura

CU01.2 - 1  El sistema actualiza el valor de la variable que almacena la altura al mínimo permitido según RN01 Valor permitido para la dimensión del espacio

- - - - *Fin de la trayectoria.*

Trayectoria alternativa CU01.3:

Condición: Ingresó un valor incorrecto para la anchura

CU01.3 - 1  El sistema actualiza el valor de la variable que almacena la anchura al mínimo permitido según RN01 Valor permitido para la dimensión del espacio

- - - - *Fin de la trayectoria.*

4.5.2. CU02 Definir tipo de vecindad

Descripción completa

Los AC bidimensionales trabajan principalmente sobre dos tipos de vecindad: la de Moore y la de von Neumann. La vecindad de Moore está definida por la célula central $c_{i,j}$ y sus vecinos $c_{i-1,j-1}, c_{i,j-1}, c_{i+1,j-1}, c_{i-1,j}, c_{i+1,j}, c_{i-1,j+1}, c_{i,j+1}, c_{i+1,j+1}$. Mientras que la vecindad de von Neumann está definida por la célula central $c_{i,j}$ y sus vecinos $c_{i,j-1}, c_{i-1,j}, c_{i+1,j}, c_{i,j+1}$.

Atributos importantes

Caso de Uso:	CU02 Definir tipo de vecindad
Versión:	1.2
Actor:	Usuario
Propósito:	Definir el tipo de vecindad para evaluar el AC
Resumen:	El usuario indica el tipo de vecindad que usará el sistema para evaluar cada célula en el autómata
Entradas:	Estado del tipo de vecindad
Salidas:	Ninguna
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Esquivel Valdez Alberto
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

1  Presiona  **Moore** vía  IU-CU02 Definir tipo de vecindad como Moore [Trayectoria Alternativa CU02.1]

2  El sistema actualiza el valor de la variable que almacena el tipo de vecindad a verdadero
- - - - *Fin del caso de uso.*

Trayectoria alternativa CU02.1:

Condición: Definir tipo de vecindad como von Neumann

CU02.1 - 1 ⚙ Presiona von Neumann vía IU-CU02.1 Definir tipo de vecindad como von Neumann

CU02.1 - 2 ○ El sistema actualiza el valor de la variable que almacena el tipo de vecindad a falso
- - - *Fin de la trayectoria.*

4.5.3. CU03 Definir tipo de regla

Descripción completa

Los AC bidimensionales trabajan principalmente sobre la regla del tipo totalista, que considera las células que se encuentran dentro de la vecindad definida (Moore o von Neumann), y la semitotalista, que además considera la célula central para realizar la evaluación de la siguiente generación.

Atributos importantes

Caso de Uso:	CU03 Definir tipo de regla
Versión:	1.2
Actor:	Usuario
Propósito:	Definir el tipo de regla para evaluar el AC
Resumen:	El usuario indica el tipo de regla que usará el sistema para evaluar cada célula en el autómata, la cual puede ser totalista o semitotalista
Entradas:	Estado del tipo de regla
Salidas:	Ninguna
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Esquivel Valdez Alberto
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 ⚙ Presiona Totalistic rules vía IU-CU03 Definir tipo de regla como totalista [Trayectoria Alternativa CU03.1]
- 2 ○ El sistema actualiza el valor de la variable que almacena el tipo de regla a totalista (verdadero)
- - - *Fin del caso de uso.*

Trayectoria alternativa CU03.1:

Condición: El valor del tipo de regla es semitotalista

CU03.1 - 1 ○ El sistema actualiza el valor de la variable que almacena el tipo de regla a semitotalista (falso)
- - - *Fin de la trayectoria.*

4.5.4. CU04 Definir los valores de la regla totalista

Descripción completa

En un AC se debe calcular la i-ésima generación dependiendo de las reglas establecidas para decidir si la célula $c_{i,j}$ en la generación $t + 1$ vive o muere. Para las reglas totalistas se utiliza una serie de configuraciones en las que el usuario determina para cada una si la célula en cuestión vive o muere.

Atributos importantes

Caso de Uso:	CU04 Definir los valores de la regla totalista
Versión:	1.2
Actor:	Usuario
Propósito:	Definir los valores que tendrán las células de acuerdo con las 32 configuraciones disponibles para las reglas totalistas
Resumen:	El usuario cambia el valor de las variables que almacenan las condiciones de supervivencia o nacimiento de la célula $c_{i,j}$
Entradas:	Valores de las células para cada configuración, 0 o 1 (viva o muerta)
Salidas:	ID de la regla totalista
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Esquivel Valdez Alberto
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 ⚙ Presiona **Valor de la célula** vía IU-CU04 Cambiar valor de la célula para una determinada configuración [Trayectoria Alternativa CU04.1] o [Trayectoria Alternativa 04.2]
 - 2 ○ El sistema obtiene el valor de la célula para la configuración previamente elegida
 - 3 ○ El sistema actualiza el valor de la célula por su complemento a uno
 - 4 ○ El sistema establece el nuevo valor de la célula para la configuración elegida
 - 5 ○ El sistema obtiene los valores de cada configuración
 - 6 ○ El sistema calcula el ID de la regla de acuerdo con RN02 ID de la regla totalista
 - 7 ○ El sistema muestra el ID de la regla totalista
- *Fin del caso de uso.*

Trayectoria alternativa CU04.1:

Condición: Ingresa el valor de la variable de densidad vía IU-CU04.1 Cambiar la densidad para la regla totalista aleatoria

- CU04.1 - 1** ○ El sistema valida que el valor ingresado sea correcto con base en RN03 Valor permitido para definir la densidad de células vivas para la regla totalista
- CU04.1 - 2** ○ El sistema actualiza el valor de la variable que almacena la densidad con el ingresado por el usuario
- CU04.1 - 3** ⚙ Presiona **Random rule** vía IU-CU04.1-1 Establecer regla aleatoria
- CU04.1 - 4** ○ El sistema actualiza los valores de las células para cada una de las 32 configuraciones disponibles de acuerdo con la densidad establecida previamente
- CU04.1 - 5** ○ El sistema calcula el ID de la regla de acuerdo con RN02 ID de la regla totalista
- CU04.1 - 6** ○ El sistema muestra el ID de la regla totalista
- *Fin de la trayectoria.*

Trayectoria alternativa CU04.2:

Condición: Ingresa el valor del ID de la regla totalista vía IU-CU04.2 Ingresar ID de la regla totalista

- CU04.2 - 1** ○ El sistema calcula los valores de la regla totalista de acuerdo con RN04 Valores de regla totalista
- CU04.2 - 2** ○ El sistema actualiza los valores de la regla
- CU04.2 - 3** ○ El sistema muestra los valores de la regla totalista
- - - - *Fin de la trayectoria.*

4.5.5. CU05 Restablecer valores de la regla totalista

Descripción completa

Atributos importantes

Caso de Uso:	CU05 Restablecer valores de la regla totalista
Versión:	1.2
Actor:	Usuario
Propósito:	Restablecer los valores de la regla totalista
Resumen:	El usuario puede restablecer los valores de la regla totalista
Entradas:	Ninguna
Salidas:	Ninguna
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Esquivel Valdez Alberto
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 ⌂ Presiona **Clear** vía  IU-CU05 Restablecer valores de la regla totalista
 - 2 ○ El sistema establece cada valor de las células como muerta (0)
 - 3 ○ El sistema calcula el ID de la regla de acuerdo con RN02 ID de la regla totalista
 - 4 ○ El sistema muestra el ID de la regla totalista
- - - - *Fin del caso de uso.*

4.5.6. CU06 Definir los valores de la regla semitotalista

Descripción completa

En un AC se debe calcular la i-ésima generación dependiendo de las reglas establecidas para decidir si la célula $c_{i,j}$ en la generación $t + 1$ vive o muere. Si la célula $c_{i,j}$ está viva y el número de vecinos vivos de dicha célula están entre los valores de las variables S_{min} y S_{max} entonces esta célula para la generación $t + 1$ vive, si la célula $c_{i,j}$ está muerta y el número de vecinos vivos alrededor de la célula están entre los valores de las variables N_{min} y N_{max} entonces dicha célula para la generación $t + 1$ vive, en cualquier otro caso la célula muere o se mantiene inactiva.

Atributos importantes

Caso de Uso:	CU06 Definir los valores de la regla semitotalista
Versión:	1.2
Actor:	Usuario
Propósito:	Definir los valores que tendrán las variables N_{min} , N_{max} , S_{min} , S_{max}
Resumen:	El usuario cambia el valor de las variables que almacenan las condiciones de supervivencia o nacimiento de la célula $c_{i,j}$
Entradas:	N_{min} , N_{max} , S_{min} , S_{max}
Salidas:	Ninguna
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Vargas Romero Erick Efraín
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 Ingresa el valor de la variable N_{min} vía IU-CU06 Definir las reglas de evaluación del AC [Trayectoria Alternativa CU06.1] o [Trayectoria Alternativa CU06.2] o [Trayectoria Alternativa CU06.3]
- 2 El sistema valida que el valor ingresado por el usuario sea correcto con base en RN05 Valor permitido para las variables condición de nacimiento [Trayectoria Alternativa CU06.4]
- 3 El sistema actualiza el valor de la variable que almacena N_{min} con el ingresado por el usuario
- - - Fin del caso de uso.

Trayectoria alternativa CU06.1:

Condición: Ingresar valor de la variable N_{max}

- CU06.1 - 1** Ingresa el valor de la variable N_{max} vía IU-CU06.1 Definir las reglas de evaluación del AC
- CU06.1 - 2** El sistema valida que el valor ingresado por el usuario sea correcto con base en RN05 Valor permitido para las variables condición de nacimiento [Trayectoria Alternativa CU06.5]
- CU06.1 - 3** El sistema actualiza el valor de la variable que almacena N_{max} con el ingresado por el usuario
- - - Fin de la trayectoria.

Trayectoria alternativa CU06.2:

Condición: Ingresar valor de la variable S_{min}

- CU06.2 - 1** Ingresa el valor de la variable S_{min} vía IU-CU06.2 Definir las reglas de evaluación del AC
- CU06.2 - 2** El sistema valida que el valor ingresado por el usuario sea correcto con base en RN05 Valor permitido para las variables condición de supervivencia [Trayectoria Alternativa CU06.6]
- CU06.2 - 3** El sistema actualiza el valor de la variable S_{min} con el ingresado por el usuario
- - - Fin de la trayectoria.

Trayectoria alternativa CU06.3:

Condición: Ingresar valor de la variable S_{max}

- CU06.3 - 1** Ingresa el valor de la variable S_{max} vía IU-CU06.3 Definir las reglas de evaluación del AC

CU06.3 - 2 ○ El sistema valida que el valor ingresado por el usuario sea correcto con base en RN05 Valor permitido para las variables condición de supervivencia [Trayectoria Alternativa CU06.7]

CU06.3 - 3 ○ El sistema actualiza el valor de la variable S_{max} con el ingresado por el usuario
- - - - Fin de la trayectoria.

Trayectoria alternativa CU06.4:

Condición: Ingresó un valor incorrecto en el campo N_{min}

CU06.4 - 1 ○ El sistema actualiza el valor de N_{min} al máximo permitido según los valores ingresados previamente por el usuario

- - - - Fin de la trayectoria.

Trayectoria alternativa CU06.5:

Condición: Ingresó un valor incorrecto en el campo N_{max}

CU06.5 - 1 ○ El sistema actualiza el valor de N_{max} al máximo permitido según los valores ingresados previamente por el usuario

- - - - Fin de la trayectoria.

Trayectoria alternativa CU06.6:

Condición: Ingresó un valor incorrecto en el campo S_{min}

CU06.6 - 1 ○ El sistema actualiza el valor de S_{min} al máximo permitido según los valores ingresados previamente por el usuario

- - - - Fin de la trayectoria.

Trayectoria alternativa CU06.7:

Condición: Ingresó un valor incorrecto en el campo S_{max}

CU06.7 - 1 ○ El sistema actualiza el valor de S_{max} al máximo permitido según los valores ingresados previamente por el usuario

- - - - Fin de la trayectoria.

4.5.7. CU07 Definir tipo de espacio

Descripción completa

Atributos importantes

Caso de Uso:	CU07 Definir tipo de espacio
Versión:	1.2
Actor:	Usuario
Propósito:	Definir el tipo de espacio de evolución del AC
Resumen:	El usuario indica el tipo de espacio que usará el sistema para evaluar las células en el AC, este puede ser abierto o cerrado (se simula el comportamiento de un toroide)
Entradas:	Ninguna
Salidas:	Ninguna
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Vargas Romero Erick Efraín
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1  Presiona Closed space vía  IU-CU07 Definir tipo de espacio como cerrado [Trayectoria Alternativa CU07.1]
- 2  El sistema actualiza el valor de la variable que almacena el tipo de espacio a verdadero
--- - Fin del caso de uso.

Trayectoria alternativa CU07.1:

Condición: Presiona Open space vía  IU-CU07.1 Definir tipo de espacio como abierto

- CU07.1 - 1**  El sistema actualiza el valor de la variable que almacena el tipo de espacio a falso
--- - Fin de la trayectoria.

4.5.8. CU08 Establecer configuración inicial del AC

Descripción completa

Atributos importantes

Caso de Uso:	CU08 Establecer configuración inicial del AC
Versión:	1.2
Actor:	Usuario
Propósito:	Definir el tipo de configuración que se utilizará para inicializar el AC
Resumen:	El usuario tiene la posibilidad de elegir la configuración inicial para el AC, ya sea mediante una densidad para las células vivas o cargando un archivo con una configuración determinada.
Entradas:	Densidad o archivo con la configuración inicial
Salidas:	Ninguna
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Esquivel Valdez Alberto
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1  Presiona Random vía  IU-CU08 Establecer configuración inicial aleatoria [Trayectoria Alternativa CU08.1]
- 2  El sistema actualiza la variable que almacena el tipo de configuración inicial a aleatorio
- 3  El sistema habilita el campo para ingresar la densidad
- 4  Ingresa el valor de la variable densidad
- 5  El sistema valida que el valor ingresado sea correcto con base en RN06 Valor permitido para definir la densidad de aparición de las células vivas [Trayectoria Alternativa CU08.2]
- 6  El sistema actualiza el valor de la variable que almacena la densidad con el ingresado por el usuario

- - - - Fin del caso de uso.

Trayectoria alternativa CU08.1:

Condición: Presiona From file vía IU-CU08.1 Establecer configuración inicial desde archivo

- CU08.1 - 1** El sistema actualiza la variable que almacena el tipo de configuración inicial a *desde archivo*
- CU08.1 - 2** El sistema deshabilita el campo para ingresar la densidad
- CU08.1 - 3** Presiona Import CA vía IU-CU08.1-1 Importar archivo
- CU08.1 - 4** El sistema muestra una ventana de selección donde están todos los directorios y archivos existentes en la computadora
- CU08.1 - 5** Selecciona el archivo que desea que sea cargado en el sistema vía IU-CU08.1-2 Seleccionar archivo [Trayectoria Alternativa CU08.3]
- CU08.1 - 6** El sistema muestra el nombre del archivo seleccionado
- CU08.1 - 7** Presiona Aceptar [Trayectoria Alternativa CU08.4]
- CU08.1 - 8** El sistema obtiene el objeto del tipo *Cellular automata* que contiene toda la información del AC

- - - - Fin de la trayectoria.

Trayectoria alternativa CU08.2:

Condición: Ingresó un valor incorrecto para la densidad

- CU08.2 - 1** El sistema actualiza el valor de la variable que almacena la densidad al máximo permitido según el valor ingresado previamente por el usuario

- - - - Fin de la trayectoria.

Trayectoria alternativa CU08.3:

Condición: No seleccionó ningún archivo

- CU08.3 - 1** Presiona Aceptar
- CU08.3 - 2** El sistema cierra la ventana de selección de archivos

- - - - Fin de la trayectoria.

Trayectoria alternativa CU08.4:

Condición: Presiona Cancelar

- CU08.4 - 1** El sistema cierra la ventana de selección de archivos

- - - - Fin de la trayectoria.

4.5.9. CU09 Exportar el objeto “Cellular automata”

Descripción completa

Atributos importantes

Caso de Uso:	CU09 Exportar el objeto “Cellular automata”
Versión:	1.2
Actor:	Usuario

Caso de Uso:	CU09 Exportar el objeto “Cellular automata”
Propósito:	Guardar la configuración actual del simulador para poder utilizarla en otro momento
Resumen:	El usuario tiene la posibilidad de exportar la configuración actual del simulador
Entradas:	Directorio al que se exporta el archivo
Salidas:	Archivo binario con extensión .dat
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Esquivel Valdez Alberto
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 ⚡ Presiona **Export CA** vía IU-CU09 Exportar el AC
- 2 ○ El sistema muestra una ventana de selección donde están todos los directorios existentes en la computadora
- 3 ⚡ Selecciona el directorio donde desea guardar el objeto “Cellular automata” vía IU-CU09.1 Seleccionar directorio
- 4 ○ El sistema muestra el nombre del directorio seleccionado
- 5 ○ El sistema establece el nombre del archivo según RN07 Nombre de archivo a exportar
- 6 ○ El sistema muestra el nombre del archivo
- 7 ⚡ Presiona **Aceptar** [Trayectoria Alternativa CU09.1]
- 8 ○ El sistema obtiene el directorio seleccionado
- 9 ○ El sistema obtiene la dimensión del espacio
- 10 ○ El sistema obtiene el tipo de regla de evaluación
- 11 ○ El sistema obtiene el tipo de vecindad
- 12 ○ El sistema obtiene el tipo de espacio
- 13 ○ El sistema obtiene la densidad de la regla totalista
- 14 ○ El sistema obtiene el escalar de proyección
- 15 ○ El sistema obtiene el tamaño de las células
- 16 ○ El sistema obtiene la velocidad de evolución
- 17 ○ El sistema obtiene la generación actual
- 18 ○ El sistema obtiene la densidad de células vivas en la generación actual
- 19 ○ El sistema obtiene la densidad de células vivas acumulada
- 20 ○ El sistema obtiene el color de las células vivas
- 21 ○ El sistema obtiene el color de las células muertas
- 22 ○ El sistema obtiene el color de las células de proyección
- 23 ○ El sistema obtiene el color de la rejilla
- 24 ○ El sistema obtiene el estado de la rejilla
- 25 ○ El sistema obtiene el estado del gradiente
- 26 ○ El sistema obtiene la regla totalista

- 27 ○ El sistema obtiene el estado de cada célula del espacio
 28 ○ El sistema obtiene la serie de puntos de la gráfica de entropía
 29 ○ El sistema obtiene la serie de puntos de la gráfica de células vivas
 30 ○ El sistema almacena los datos obtenidos en el archivo especificado previamente [Trayectoria Alternativa CU09.2]
 31 ○ El sistema muestra ✎ MSG-CU09 AC guardado correctamente
- - - - *Fin del caso de uso.*

Trayectoria alternativa CU09.1:

Condición: Presiona

- CU09.1 - 1 ○ El sistema cierra la ventana de selección de directorio
 CU09.1 - 2 ○ El sistema muestra ✎ MSG-CU09.1 Seleccionar un directorio
 - - - - *Fin de la trayectoria.*

Trayectoria alternativa CU09.2:

Condición: El archivo ya existe

- CU09.2 - 1 ○ El sistema muestra ✎ MSG-CU09.2 ¿Desea sobreescribir el archivo?
 CU09.2 - 2 ⚙ Presiona
 CU09.2 - 3 ○ El sistema sobreescribe el contenido del archivo con los datos obtenidos previamente
 CU09.2 - 4 ○ El sistema muestra ✎ MSG-CU09.3 AC guardado correctamente
 - - - - *Fin de la trayectoria.*

4.5.10. CU10 Definir tamaño de las células

Descripción completa

Al definir un tamaño de células grande (mayor o igual a 5) el usuario podrá ver con mayor detalle la generación actual en una región específica y si establece un tamaño pequeño (menor que 5) tendrá la posibilidad de ver un mayor número de células sobre el espacio.

Atributos importantes

Caso de Uso:	CU10 Definir tamaño de las células
Versión:	1.2
Actor:	Usuario
Propósito:	Visualizar mejor la generación actual del AC
Resumen:	El usuario ingresa el tamaño de las células en pixeles. Las células son cuadradas, por lo que el usuario solo especifica el valor de uno de los lados.
Entradas:	Número de pixeles por célula
Salidas:	Ninguna
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Esquivel Valdez Alberto
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 Ingresa el número de pixeles por célula vía IU-CU10 Definir tamaño de las células
 - 2 El sistema valida que el valor ingresado sea correcto con base en RN08 Valor permitido para tamaño de las células [Trayectoria Alternativa CU10.1]
 - 3 El sistema actualiza el valor de la variable que almacena el tamaño de célula con el número de pixeles ingresado por el usuario
- - - - *Fin del caso de uso.*

Trayectoria alternativa CU10.1:

Condición: Ingresó un valor incorrecto en el campo de tamaño de célula

- CU10.1 - 1** El sistema actualiza el valor de la variable que almacena el tamaño de la célula al máximo permitido según RN08 Valor permitido para tamaño de las células

- - - - *Fin de la trayectoria.*

4.5.11. CU11 Definir la velocidad de evolución del AC

Descripción completa

Atributos importantes

Caso de Uso:	CU11 Definir la velocidad de evolución del AC
Versión:	1.2
Actor:	Usuario
Propósito:	Definir la velocidad de evolución entre cada generación
Resumen:	El usuario puede elegir la velocidad en milisegundos con la que evolucionará el AC en las siguientes generaciones
Entradas:	Valor de la velocidad en milisegundos
Salidas:	Ninguna
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Vargas Romero Erick Efraín
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 Ingresa el valor en milisegundos para la variable de velocidad de evolución vía IU-CU11 Definir la velocidad de evolución del AC
 - 2 El sistema valida que el valor ingresado sea correcto según RN09 Valor permitido para el tiempo de evolución [Trayectoria Alternativa CU11.1]
 - 3 El sistema actualiza el valor de la variable que almacena la velocidad de evolución con el ingresado por el usuario
- - - - *Fin del caso de uso.*

Trayectoria alternativa CU11.1:

Condición: Ingresó un valor incorrecto para la velocidad

- CU11.1 - 1** ○ El sistema actualiza el valor de la variable que almacena la velocidad al máximo permitido según el valor ingresado previamente por el usuario

- - - - *Fin de la trayectoria.*

4.5.12. CU12 Definir el escalar de proyección

Descripción completa

El simulador da la posibilidad de realizar una proyección a tres dimensiones, para realizar esto es necesario definir una submatriz que evolucionará a través del tiempo (el espacio de evolución normal para el AC). El resto del espacio de evolución se desplazará de forma diagonal hacia el nor-este para simular la proyección. Las células que corresponden a la proyección no evolucionan a través del tiempo.

Atributos importantes

Caso de Uso:	CU12 Definir el escalar de proyección
Versión:	1.1
Actor:	Usuario
Propósito:	
Resumen:	El usuario define el valor de un número escalar que permita obtener una submatriz del espacio de evolución
Entradas:	Valor del escalar
Salidas:	Ninguna
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Vargas Romero Erick Efraín
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 ⚙ Ingresa el valor que tendrá el escalar vía IU-CU12 Asignar valor de escalar
 - 2 ○ El sistema valida que el valor ingresado sea correcto con base en RN10 Valor permitido para el escalar de proyección [Trayectoria Alternativa CU12.1]
 - 3 ○ El sistema actualiza el valor de la variable que almacena el escalar de proyección con el ingresado por el usuario
- - - - *Fin del caso de uso.*

Trayectoria alternativa CU12.1:

Condición: Ingresó un valor incorrecto para el escalar de proyección

- CU12.1 - 1** ○ El sistema actualiza el valor de la variable que almacena el escalar de proyección al máximo permitido según RN10 Valor permitido para el escalar de proyección

- - - - *Fin de la trayectoria.*

4.5.13. CU13 Establecer el estado del gradiente

Descripción completa

Atributos importantes

Caso de Uso:	CU13 Establecer el estado del gradiente
Versión:	1.2
Actor:	Usuario
Propósito:	Mostrar u ocultar las células que han sobrevivido durante varias generaciones.
Resumen:	El usuario tiene la posibilidad de elegir si en el espacio de evoluciones se mostrarán con otro color las células que han sobrevivido durante varias generaciones.
Entradas:	Estado del gradiente
Salidas:	Ninguna
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Esquivel Valdez Alberto
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 Presiona Gradient vía IU-CU13 Estado del gradiente [Trayectoria Alternativa CU13.1]
 - 2 El sistema actualiza la variable que almacena el estado del gradiente a verdadero
 - 3 El sistema obtiene el color de las células vivas con gradiente según RN11 Color de células con gradiente
 - 4 El sistema verifica que exista un objeto del tipo “Cellular automata”[Trayectoria Alternativa CU13.2]
 - 5 El sistema actualiza la variable que almacena el color de las células vivas con el seleccionado por el usuario
 - 6 El sistema actualiza el color de las células que han sobrevivido en el espacio de evolución con el valor obtenido
- - - - *Fin del caso de uso.*

Trayectoria alternativa CU13.1:

Condición: El valor actual del estado del gradiente es verdadero

- CU13.1 - 1** El sistema actualiza la variable que almacena el estado del gradiente a falso
 - CU13.1 - 2** El sistema obtiene el color de las células vivas
 - CU13.1 - 3** El sistema verifica que exista un objeto del tipo “Cellular automata”[Trayectoria Alternativa CU13.2]
 - CU13.1 - 4** El sistema actualiza la variable que almacena el color de las células vivas con el seleccionado por el usuario
 - CU13.1 - 5** El sistema actualiza el color de las células vivas en el espacio de evolución con el valor obtenido
- - - - *Fin de la trayectoria.*

Trayectoria alternativa CU13.2:

Condición: No existe el objeto “Cellular automata”

- CU13.2 - 1** El sistema actualiza la variable que almacena el color de las células vivas con el seleccionado por el usuario
- - - - *Fin de la trayectoria.*

4.5.14. CU14 Establecer el estado de la rejilla

Descripción completa

Atributos importantes

Caso de Uso:	CU14 Establecer el estado de la rejilla
Versión:	1.2
Actor:	Usuario
Propósito:	Mostrar u ocultar la rejilla para visualizar mejor la generación actual del AC
Resumen:	El usuario tiene la posibilidad de elegir si en el espacio de evoluciones se mostrarán las divisiones entre células (la rejilla).
Entradas:	Estado de la rejilla
Salidas:	Ninguna
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Vargas Romero Erick Efraín
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 ⌂ Presiona Grid vía IU-CU14 Estado de la rejilla [Trayectoria Alternativa CU14.1]
 - 2 ⌂ El sistema actualiza la variable que almacena el estado de la rejilla a verdadero
 - 3 ⌂ El sistema renderiza el espacio de evolución vía IU-CU33.3 Espacio de evolución con la rejilla
- - - - *Fin del caso de uso.*

Trayectoria alternativa CU14.1:

Condición: El valor actual del estado de la rejilla es verdadero

CU14.1 - 1 ⌂ El sistema actualiza la variable que almacena el estado de la rejilla a falso

CU14.1 - 2 ⌂ El sistema renderiza el espacio de evolución vía IU-CU14-1 Espacio de evolución sin la rejilla

- - - - *Fin de la trayectoria.*

4.5.15. CU15 Definir el color de la rejilla

Descripción completa

Atributos importantes

Caso de Uso:	CU15 Definir el color de la rejilla
Versión:	1.1
Actor:	Usuario
Propósito:	Definir el color de la rejilla que divide las células del espacio de evolución
Resumen:	El usuario ingresa el color de la rejilla mediante un selector de color que tenga los campos correspondientes para ingresar el valor en formato hexadecimal y en RGB
Entradas:	Valor del color de la célula en formato hexadecimal o RGB

Caso de Uso:	CU15 Definir el color de la rejilla
Salidas:	Ninguna
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Vargas Romero Erick Efraín
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 ⚒ Presiona **Grid lines** vía IU-CU15 Definir el color de la rejilla
 - 2 ○ El sistema despliega un cuadro de diálogo con la paleta de colores
 - 3 ⚒ Ingresá el color de la rejilla vía IU-CU15-1 Ventana de selección de color
 - 4 ○ El sistema valida que el valor ingresado sea correcto con base en la RN12 Valor permitido para el color de la rejilla[Trayectoria Alternativa CU15.1]
 - 5 ⚒ Presiona **Aceptar**[Trayectoria Alternativa CU15.2]
 - 6 ○ El sistema actualiza el valor de la variable que almacena el color de la rejilla con el ingresado por el usuario
 - 7 ○ El sistema actualiza el color de la etiqueta indicadora de la rejilla vía IU-CU15.1 Etiqueta indicadora de la rejilla
 - 8 ○ El sistema verifica que exista un objeto del tipo “Cellular automata”[Trayectoria Alternativa CU15.3]
 - 9 ○ El sistema actualiza la variable que almacena el color de la rejilla con el seleccionado por el usuario
 - 10 ○ El sistema renderiza el espacio de evolución vía IU-CU33.3 Espacio de evolución
- *Fin del caso de uso.*

Trayectoria alternativa CU15.1:

Condición: Ingresó un valor incorrecto para el color de la rejilla

CU15.1 - 1 ⚒ Presiona **Aceptar**

CU15.1 - 2 ○ El sistema cierra el cuadro de diálogo con la paleta de colores

--- *Fin de la trayectoria.*

Trayectoria alternativa CU15.2:

Condición: Presiona **Cancelar**

CU15.2 - 1 ○ El sistema cierra el cuadro de diálogo con la paleta de colores

--- *Fin de la trayectoria.*

Trayectoria alternativa CU15.3:

Condición: No existe el objeto “Cellular automata”

CU15.3 - 1 ○ El sistema actualiza la variable que almacena el color de la rejilla con el seleccionado por el usuario

--- *Fin de la trayectoria.*

4.5.16. CU16 Definir color de las células

Descripción completa

Para poder diferenciar las células vivas de las células muertas y de las células de proyección el usuario puede definir el color que las representa.

Atributos importantes

Caso de Uso:	CU16 Definir color de las células
Versión:	1.2
Actor:	Usuario
Propósito:	Diferenciar las células vivas, muertas y de proyección mediante un color para cada tipo.
Resumen:	El usuario ingresa el color de las células mediante un selector de color que tenga los campos correspondientes para ingresar el valor en formato hexadecimal y en RGB.
Entradas:	Valor del color de la célula en formato hexadecimal o RGB
Salidas:	Ninguna
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Esquivel Valdez Alberto
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 Presiona **Living cells** vía IU-CU16 Definir color de las células vivas [Trayectoria Alternativa CU16.1] o [Trayectoria Alternativa CU16.2]
 - 2 El sistema despliega un cuadro de diálogo con la paleta de colores
 - 3 Ingresa el color de las células vivas vía IU-CU15-1 Ventana de selección de color
 - 4 El sistema valida que el valor ingresado sea correcto con base en RN12 Valor permitido para el color de la célula [Trayectoria Alternativa CU16.3]
 - 5 Presiona **Aceptar** [Trayectoria Alternativa CU16.6]
 - 6 El sistema actualiza el valor de la variable que almacena el color de células vivas con el ingresado el usuario
 - 7 El sistema actualiza el color de la etiqueta indicadora de las células vivas vía IU-CU16.3 Etiqueta indicadora de las células vivas
 - 8 El sistema verifica que exista un objeto del tipo “Cellular automata”[Trayectoria Alternativa CU16.7]
 - 9 El sistema actualiza la variable que almacena el color de las células vivas con el seleccionado por el usuario
 - 10 El sistema renderiza el espacio de evolución vía IU-CU16-1 Espacio de evolución con el nuevo color de células vivas
- - - *Fin del caso de uso.*

Trayectoria alternativa CU16.1:

Condición: Ingresar el color de las células muertas

- CU16.1 - 1** ⚒ Presiona **Death cells** vía IU-CU16.1 Definir color de las células muertas
- CU16.1 - 2** ○ El sistema despliega un cuadro de diálogo con la paleta de colores
- CU16.1 - 3** ⚒ Ingresa el color de las células muertas vía IU-CU15-1 Ventana de selección de color
- CU16.1 - 4** ○ El sistema valida que el valor ingresado sea correcto con base en RN12 Valor permitido para el color de la célula [Trayectoria Alternativa CU16.4]
- CU16.1 - 5** ⚒ Presiona **Aceptar** [Trayectoria Alternativa CU16.6]
- CU16.1 - 6** ○ El sistema actualiza el valor de la variable que almacena el color de células muertas con el ingresado el usuario
- CU16.1 - 7** ○ El sistema actualiza el color de la etiqueta indicadora de las células muertas vía IU-CU16.3 Etiqueta indicadora de las células muertas
- CU16.1 - 8** ○ El sistema verifica que exista un objeto del tipo “Cellular automata”[Trayectoria Alternativa CU16.7]
- CU16.1 - 9** ○ El sistema actualiza la variable que almacena el color de las células muertas con el seleccionado por el usuario
- CU16.1 - 10** ○ El sistema renderiza el espacio de evolución vía IU-CU16-2 Espacio de evolución con el nuevo color de células muertas
- - - *Fin de la trayectoria.*

Trayectoria alternativa CU16.2:

Condición: Ingresar el color de las células de proyección

- CU16.2 - 1** ⚒ Presiona **Projection cells** vía IU-CU16.2 Definir color de las células de proyección
- CU16.2 - 2** ○ El sistema despliega un cuadro de diálogo con la paleta de colores
- CU16.2 - 3** ⚒ Ingresa el color de las células de proyección vía IU-CU15-1 Ventana de selección de color
- CU16.2 - 4** ○ El sistema valida que el valor ingresado sea correcto con base en RN12 Valor permitido para el color de la célula [Trayectoria Alternativa CU16.5]
- CU16.2 - 5** ⚒ Presiona **Aceptar** [Trayectoria Alternativa CU16.6]
- CU16.2 - 6** ○ El sistema actualiza el valor de la variable que almacena el color de células de proyección con el ingresado el usuario
- CU16.2 - 7** ○ El sistema actualiza el color de la etiqueta indicadora de las células de proyección vía IU-CU16.3 Etiqueta indicadora de las células de proyección
- CU16.2 - 8** ○ El sistema verifica que exista un objeto del tipo “Cellular automata”[Trayectoria Alternativa CU16.7]
- CU16.2 - 9** ○ El sistema actualiza la variable que almacena el color de las células de proyección con el seleccionado por el usuario
- CU16.2 - 10** ○ El sistema renderiza el espacio de evolución vía IU-CU16-3 Espacio de evolución con el nuevo color de células de proyección
- - - *Fin de la trayectoria.*

Trayectoria alternativa CU16.3:

Condición: Ingresó un valor incorrecto para el color de las células vivas

- CU16.3 - 1** ⚒ Presiona **Aceptar**
- CU16.3 - 2** ○ El sistema cierra el cuadro de diálogo con la paleta de colores
- - - *Fin de la trayectoria.*

Trayectoria alternativa CU16.4:

Condición: Ingresó un valor incorrecto para el color de las células muertas

CU16.4 - 1  Presiona **Aceptar**

CU16.4 - 2  El sistema cierra el cuadro de diálogo con la paleta de colores

- - - - *Fin de la trayectoria.*

Trayectoria alternativa CU16.5:

Condición: Ingresó un valor incorrecto para el color de las células de proyección

CU16.5 - 1  Presiona **Aceptar**

CU16.5 - 2  El sistema cierra el cuadro de diálogo con la paleta de colores

- - - - *Fin de la trayectoria.*

Trayectoria alternativa CU16.6:

Condición: Presiona **Cancelar**

CU16.6 - 1  El sistema cierra el cuadro de diálogo con la paleta de colores

- - - - *Fin de la trayectoria.*

Trayectoria alternativa CU16.7:

Condición: No existe el objeto “Cellular automata”

CU16.7 - 1  El sistema actualiza la variable que almacena el color de las células con el seleccionado por el usuario

- - - - *Fin de la trayectoria.*

4.5.17. CU17 Cambiar el estado de una célula

Descripción completa

Atributos importantes

Caso de Uso:	CU17 Cambiar el estado de una célula
Versión:	1.2
Actor:	Usuario
Propósito:	Cambiar el estado de la célula $c_{i,j}$ del espacio de evolución
Resumen:	El usuario da click sobre la célula $c_{i,j}$ del espacio de evolución y está cambiando dependiendo del estado actual, si la célula posee el estado de viva esta cambia a muerta y viceversa.
Entradas:	Coordenada de la célula $c_{i,j}$, tamaño de la célula
Salidas:	Ninguna
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Vargas Romero Erick Efraín
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 Presiona una célula del espacio de evolución vía IU-CU17 Espacio de evolución
 - 2 El sistema obtiene la coordenada del evento click
 - 3 El sistema obtiene la variable que posee el tamaño de las células
 - 4 El sistema obtiene la posición de la célula $c_{i,j}$ con base en la RN13 Posición de una célula
 - 5 El sistema actualiza el estado la célula $c_{i,j}$ por su complemento a uno
 - 6 El sistema renderiza el espacio de evolución vía IU-CU17 Espacio de evolución
- *Fin del caso de uso.*

4.5.18. CU18 Desplegar ventana de configuración

Descripción completa

Atributos importantes

Caso de Uso:	CU18 Desplegar ventana de configuración
Versión:	1.2
Actor:	Usuario
Propósito:	Mostrar al usuario la ventana donde configurará el AC
Resumen:	El sistema cuenta con una ventana de configuración para que el usuario pueda ingresar y elegir todas las características que tendrá el AC, así como los botones de acción
Entradas:	Ninguna
Salidas:	Ventana de configuración
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Esquivel Valdez Alberto
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 Presiona dos veces el ícono “vNCASimulator” que se encuentra en su computadora
 - 2 El sistema crea el objeto de tipo “Settings” en memoria
 - 3 El sistema despliega IU-CU18 Ventana de configuración
 - 4 El sistema renderiza IU-CU18 Ventana de configuración
- *Fin del caso de uso.*

4.5.19. CU19 Desplegar el AC con la configuración inicial

Descripción completa

Atributos importantes

Caso de Uso:	CU19 Desplegar el AC con la configuración inicial
Versión:	1.2
Actor:	Usuario
Propósito:	Mostrar al usuario la configuración inicial del AC, es decir, la generación 0
Resumen:	Una vez que el usuario ha ingresado la configuración inicial para el AC, el usuario tiene la posibilidad de solicitar que se despliegue el espacio de evolución
Entradas:	Ninguna
Salidas:	Espacio de evolución con la configuración inicial
Precondiciones:	Debe existir una configuración inicial
Postcondiciones:	Ninguna
Autor:	Esquivel Valdez Alberto
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 ⚙ Presiona  IU-CU19 Desplegar CA con la configuración inicial
 - 2 ⚙ El sistema obtiene el tipo de espacio
 - 3 ⚙ El sistema obtiene la configuración inicial
 - 4 ⚙ El sistema obtiene las dimensiones del espacio de evolución
 - 5 ⚙ El sistema obtiene el tamaño de las células
 - 6 ⚙ El sistema crea un objeto del tipo “Cellular automata” con los parámetros obtenidos
 - 7 ⚙ El sistema despliega  IU-CU19.1 Espacio de evolución
 - 8 ⚙ El sistema renderiza  IU-CU19.1 Espacio de evolución con la configuración inicial
- Fin del caso de uso.

4.5.20. CU20 Calcular siguiente generación

Descripción completa

El cálculo de la siguiente generación se realiza utilizando el espacio de evolución del AC para evaluar los vecinos de cada célula $c_{i,j}$. Esta evaluación cambia dependiendo del tipo de vecindad, tipo de espacio y tipo de regla. Si la vecindad es de tipo *Moore* se evalúan los ocho vecinos alrededor de la célula $c_{i,j}$, pero si el tipo de vecindad es de *von Neumann* entonces se evaluarán solo los vecinos inferior, superior, izquierdo y derecho. Además, si el tipo de regla es semitotalista, se considerará el valor de la célula $c_{i,j}$ para la evaluación. Por otra parte, si el tipo de espacio es cerrado se simula un *toroide* al evaluar la célula $c_{i,j}$, esto se logra haciendo uso de aritmética modular. Si el espacio es abierto, es decir *no es toroide* se realizará la evaluación como si fuese una matriz común.

Atributos importantes

Caso de Uso:	CU20 Calcular siguiente generación
Versión:	1.2
Actor:	Sistema
Propósito:	Obtener la siguiente generación del AC
Resumen:	El sistema dependiendo del tipo de vecindad y tipo de espacio definidos calcula la siguiente generación del AC
Entradas:	Espacio de evolución, tipo de vecindad, tipo de espacio
Salidas:	Siguiente generación del AC
Precondiciones:	Debe existir un objeto del tipo “Cellular automata” en memoria
Postcondiciones:	Ninguna
Autor:	Vargas Romero Erick Efraín
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 ○ El sistema obtiene el espacio de evolución
 - 2 ○ El sistema obtiene el tipo de vecindad (von Neumann o Moore)
 - 3 ○ El sistema obtiene el tipo de espacio (abierto o cerrado)
 - 4 ○ El sistema obtiene el tipo de regla
 - 5 ○ El sistema obtiene las 32 configuraciones de la regla totalista[Trayectoria Alternativa CU20.1]
 - 6 ○ El sistema evalúa el espacio de evolución según RN14 Evaluación del espacio de evolución
 - 7 ○ El sistema calcula el valor que tendrá cada célula en el tiempo $t + 1$ según RN15 Evaluación de la célula $c_{i,j}$ totalista
 - 8 ○ El sistema actualiza el valor de cada célula en una matriz auxiliar según RN16 Valor de célula
 - 9 ○ El sistema intercambia la dirección de memoria entre el espacio de evolución original y el espacio de evolución auxiliar
- - - - *Fin del caso de uso.*

Trayectoria alternativa CU20.1:

Condición: El tipo de regla es semitotalista

- CU20.1 - 1** ○ El sistema obtiene el valor de la variable S_{min}
 - CU20.1 - 2** ○ El sistema obtiene el valor de la variable S_{max}
 - CU20.1 - 3** ○ El sistema obtiene el valor de la variable N_{min}
 - CU20.1 - 4** ○ El sistema obtiene el valor de la variable N_{max}
 - CU20.1 - 5** ○ El sistema evalúa el espacio de evolución según RN14 Evaluación del espacio de evolución
 - CU20.1 - 6** ○ El sistema contabiliza el número de vecinos alrededor de la célula $c_{i,j}$ según RN17 Contabilización de los vecinos alrededor de la célula $c_{i,j}$
 - CU20.1 - 7** ○ El sistema calcula el valor que tendrá cada célula en el tiempo $t + 1$ según RN18 Evaluación de la célula $c_{i,j}$ semitotalista
 - CU20.1 - 8** ○ El sistema actualiza el valor de cada célula en una matriz auxiliar según RN16 Valor de célula
 - CU20.1 - 9** ○ El sistema intercambia la dirección de memoria entre el espacio de evolución original y el espacio de evolución auxiliar
- - - - *Fin de la trayectoria.*

4.5.21. CU21 Desplegar siguiente generación

Descripción completa

Atributos importantes

Caso de Uso:	CU21 Desplegar siguiente generación
Versión:	1.2
Actor:	Usuario
Propósito:	Visualizar la siguiente generación del AC
Resumen:	Se realiza la evaluación del espacio de evolución de acuerdo con el tipo de vecindad y tipo de espacio. Una vez calculados los valores de las células en el tiempo $t + 1$, estas son actualizadas en el simulador con su nuevo valor
Entradas:	Estado del AC en el tiempo t
Salidas:	Ninguna
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Vargas Romero Erick Efraín
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 ⚙ Presiona **Next generation** vía IU-CU21 Desplegar siguiente generación
 - 2 El sistema calcula la siguiente generación mediante CU20 Calcular siguiente generación
 - 3 El sistema obtiene el espacio de evolución para la generación $t + 1$
 - 4 El sistema realiza CU22 Calcular densidad
 - 5 El sistema realiza CU23 Calcular entropía
 - 6 El sistema renderiza IU-CU21-1 Siguiente generación del AC
- *Fin del caso de uso.*

4.5.22. CU22 Calcular densidad de población

Descripción completa

Atributos importantes

Caso de Uso:	CU22 Calcular densidad de población
Versión:	1.2
Actor:	Sistema
Propósito:	Obtener la densidad haciendo uso de la información de la generación actual
Resumen:	El sistema obtiene la densidad haciendo uso de la cantidad de células vivas en la generación actual y de células totales en el espacio de evolución
Entradas:	Número de células vivas, número total de células existentes en el espacio de evolución
Salidas:	Un número del tipo double que representa la densidad actual
Precondiciones:	Ninguna

Caso de Uso:	CU22 Calcular densidad de población
Postcondiciones:	Ninguna
Autor:	Vargas Romero Erick Efraín
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 El sistema obtiene la cantidad de células vivas en la generación actual
 - 2 El sistema obtiene la cantidad de células totales en el espacio de evolución
 - 3 El sistema realiza el cálculo de la densidad utilizando la información obtenida anteriormente según RN19 Cálculo de la densidad de población
 - 4 El sistema actualiza el valor de la variable que almacena la densidad
- - - - *Fin del caso de uso.*

4.5.23. CU23 Calcular entropía

Descripción completa

Atributos importantes

Caso de Uso:	CU23 Calcular entropía
Versión:	1.2
Actor:	Sistema
Propósito:	Obtener la entropía haciendo uso de la información de la generación actual
Resumen:	El sistema obtiene la densidad actual y realiza el cálculo de la entropía
Entradas:	Densidad
Salidas:	Un número del tipo double que representa la entropía actual
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Vargas Romero Erick Efraín
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 El sistema obtiene el valor de la densidad de células vivas en la generación actual
 - 2 El sistema realiza el cálculo de la entropía utilizando la información obtenida anteriormente según RN20 Cálculo de la entropía
 - 3 El sistema actualiza el valor de la variable que almacena la entropía
- - - - *Fin del caso de uso.*

4.5.24. CU24 Calcular polinomio característico

Descripción completa

A cada regla que el usuario define le corresponde un polinomio característico que se obtiene mediante la aplicación de la teoría del campo promedio. Es necesario calcular dicho polinomio para graficarlo y poder clasificarlo de acuerdo al comportamiento de la curva obtenida

Atributos importantes

Caso de Uso:	CU24 Calcular polinomio característico
Versión:	1.2
Actor:	Sistema
Propósito:	Calcular polinomio característico de una regla
Resumen:	El sistema obtiene los valores de $S_{min}, S_{max}, B_{min}, B_{max}$ previamente ingresados por el usuario y el tipo de vecindad que se está utilizando actualmente. Con dicha información se realiza el cálculo del polinomio característico para la regla ingresada.
Entradas:	$S_{min}, S_{max}, B_{min}, B_{max}$ y tipo de vecindad
Salidas:	Polinomio característico para la regla actual
Precondiciones:	Debe existir un objeto CA en memoria, $S_{min}, S_{max}, B_{min}, B_{max}$ deben tener valores asignados, debe estar seleccionado un tipo de vecindad
Postcondiciones:	Ninguna
Autor:	Vargas Romero Erick Efraín
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 ○ El sistema obtiene el valor de la variable S_{min}
 - 2 ○ El sistema obtiene el valor de la variable S_{max}
 - 3 ○ El sistema obtiene el valor de la variable B_{min}
 - 4 ○ El sistema obtiene el valor de la variable B_{max}
 - 5 ○ El sistema obtiene el tipo de vecindad actual
 - 6 ○ El sistema realiza el cálculo del polinomio de campo promedio utilizando los valores obtenidos anteriormente según RN21 Cálculo del polinomio característico
 - 7 ○ El sistema actualiza el valor de la variable que almacena el polinomio característico
- *Fin del caso de uso.*

4.5.25. CU25 Calcular puntos de intersección

Descripción completa

Atributos importantes

Caso de Uso:	CU25 Calcular puntos de intersección
Versión:	1.2
Actor:	Sistema

Caso de Uso:	CU25 Calcular puntos de intersección
Propósito:	Encontrar los puntos de intersección entre el polinomio característico y la recta identidad para iniciar la clasificación de las reglas
Resumen:	El programa determina los puntos de intersección entre el polinomio característico y la recta identidad
Entradas:	Ninguna
Salidas:	Ninguna
Precondiciones:	Se debe haber calculado el polinomio característico
Postcondiciones:	Ninguna
Autor:	Torres Hernández Eduardo
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 ○ El sistema obtiene el valor de la variable que almacena el polinomio característico
 - 2 ○ El sistema evalúa un millón de puntos del polinomio en el rango de $I = [0, 1]$
 - 3 ○ El sistema encuentra los puntos candidatos con base en RN22 Punto candidato [Trayectoria Alternativa CU25.1]
 - 4 ○ El sistema almacena los puntos candidatos
 - 5 ○ El sistema suma los puntos candidatos que no cumplen con una diferencia mayor a 1×10^{-6} [Trayectoria Alternativa CU25.2]
 - 6 ○ El sistema promedia la suma de los puntos candidatos para generar un punto válido con base en RN23 Punto válido para la gráfica
 - 7 ○ El sistema clasifica el punto según RN24 Tipo de punto
 - 8 ○ El sistema agrega el punto clasificado a la variable que almacena los puntos válidos
- *Fin del caso de uso.*

Trayectoria alternativa CU25.1:

Condición: El sistema no encontró puntos candidatos

- CU25.1 - 1** ○ El sistema finaliza el proceso de cálculo de puntos de intersección
 --- *Fin de la trayectoria.*

Trayectoria alternativa CU25.2:

Condición: Dos puntos no cumplen con una diferencia mayor a 1×10^{-6}

- CU25.2 - 1** ○ El sistema suma los puntos candidatos a partir del que no cumplió con la condición de diferencia
CU25.2 - 2 ○ El sistema promedia la suma de los puntos candidatos a partir del que no cumplió con la condición de diferencia para generar un punto válido con base en RN23 Punto válido para la gráfica
CU25.2 - 3 ○ El sistema clasifica el punto según RN24 Tipo de punto
CU25.2 - 4 ○ El sistema agrega el punto clasificado a la variable que almacena los puntos válidos
 --- *Fin de la trayectoria.*

4.5.26. CU26 Graficar densidad de población

Descripción completa

Atributos importantes

Caso de Uso:	CU26 Graficar densidad de población
Versión:	1.2
Actor:	Usuario
Propósito:	
Resumen:	Cuando se calcula la i-ésima generación, en cada paso se obtiene y añade un punto que será graficado, este punto representa la población de la i-ésima generación.
Entradas:	Población en la generación actual
Salidas:	Gráfica de la población
Precondiciones:	Debe existir un objeto del tipo “Cellular automata” en memoria
Postcondiciones:	Ninguna
Autor:	Vargas Romero Erick Efraín
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 ⚙ Presiona **Next generation** vía  IU-CU26 Graficar densidad de población
 - 2 ○ El sistema obtiene la cantidad de células vivas en la generación actual
 - 3 ○ El sistema agrega un punto a la gráfica de población con la cantidad obtenida
 - 4 ○ El sistema renderiza  IU-CU26.1 Gráfica de población
- *Fin del caso de uso.*

4.5.27. CU27 Graficar entropía

Descripción completa

Atributos importantes

Caso de Uso:	CU27 Graficar entropía
Versión:	1.2
Actor:	Usuario
Propósito:	
Resumen:	Cuando se calcula la i-ésima generación, en cada paso se obtiene y añade un punto el cual será graficado, este punto representa la entropía de la i-ésima generación.
Entradas:	Entropía en la generación actual
Salidas:	Gráfica de la entropía
Precondiciones:	Debe existir un objeto del tipo “Cellular automata” en memoria
Postcondiciones:	Ninguna
Autor:	Vargas Romero Erick Efraín
Estado:	Fase de pruebas
Última versión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 ⌂ Presiona **Next generation** vía IU-CU27 Graficar entropía
 - 2 ○ El sistema obtiene la entropía en la generación actual
 - 3 ○ El sistema agrega un punto a la gráfica de entropía con el dato obtenido
 - 4 ○ El sistema renderiza IU-CU27.1 Gráfica de entropía
- - - - *Fin del caso de uso.*

4.5.28. CU28 Graficar polinomio característico

Descripción completa

Atributos importantes

Caso de Uso:	CU28 Graficar polinomio característico
Versión:	1.1
Actor:	Usuario
Propósito:	Graficar el polinomio característico de una regla
Resumen:	El sistema obtiene el polinomio característico previamente calculado y realiza su evaluación en los puntos $x \in I x = \{0, 0.001, 0.002, \dots, 1\}, I = [0, 1]$. Finalmente grafica el polinomio con distancia entre cada punto de 0.001
Entradas:	Polinomio característico de la regla actual
Salidas:	Ninguna
Precondiciones:	El polinomio característico de la regla actual debe haber sido calculado
Postcondiciones:	Ninguna
Autor:	Vargas Romero Erick Efraín
Estado:	Prototipo funcional
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 ⌂ Presiona **Draw** vía IU-CU28 Graficar polinomio característico
 - 2 ○ El sistema obtiene el polinomio característico
 - 3 ○ El sistema evalúa el polinomio característico en el intervalo de $x \in I, I = [0, 1]$, tal que x toma los valores 0, 0.001, 0.002, ..., 1
 - 4 ○ El sistema agrega cada punto a la gráfica del polinomio característico
 - 5 ○ El sistema agrega el punto clasificado a la lista de puntos vía IU-CU28.1 Lista de puntos clasificados
 - 6 ○ El sistema renderiza IU-CU28.2 Gráfica del polinomio característico
- - - - *Fin del caso de uso.*

4.5.29. CU29 Graficar recta identidad

Descripción completa

Atributos importantes

Caso de Uso:	CU29 Graficar recta identidad
Versión:	1.2
Actor:	Sistema
Propósito:	
Resumen:	La recta identidad permitirá clasificar la regla que se está evaluando, de acuerdo a las intersecciones que se encuentren entre la recta y el polinomio característico obtenido mediante la aplicación de la teoría del campo promedio para la regla actual, por ello es de suma importancia mostrar esta recta para que el usuario pueda observar y comparar los resultados
Entradas:	Ninguna
Salidas:	Ninguna
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Esquivel Valdez Alberto
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1  Presiona  vía  IU-CU29 Graficar recta identidad
 - 2  El sistema agrega un punto en el par ordenado (x, x) según RN25 Puntos pertenecientes a la recta identidad
 - 3  El sistema renderiza  IU-CU29.1 Gráfica del polinomio característico con recta identidad
- Fin del caso de uso.

4.5.30. CU30 Graficar puntos de intersección

Descripción completa

Atributos importantes

Caso de Uso:	CU30 Graficar puntos de intersección
Versión:	1.2
Actor:	Usuario
Propósito:	Mostrar al usuario los puntos de intersección entre el polinomio característico y la recta identidad
Resumen:	El sistema calcula los puntos de intersección entre el polinomio característico y la recta identidad para agregar esos puntos a la gráfica correspondiente y mostrarla al usuario
Entradas:	Puntos de intersección
Salidas:	Gráfica de polinomio característico con puntos de intersección
Precondiciones:	Ninguna

Caso de Uso:	CU30 Graficar puntos de intersección
Postcondiciones:	Ninguna
Autor:	Vargas Romero Erick Efraín
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 Presiona **Draw** vía IU-CU30 Graficar puntos de intersección
 - 2 El sistema realiza CU25 Calcular puntos de intersección
 - 3 El sistema obtiene los puntos válidos de las intersecciones del tipo atractivo [Trayectoria Alternativa CU30.1] o [Trayectoria Alternativa CU30.2] o [Trayectoria Alternativa CU30.3]
 - 4 El sistema agrega los puntos del tipo atractivo a la gráfica
 - 5 El sistema renderiza IU-CU30.1 Gráfica de polinomio característico con puntos de intersección del tipo atractivo
- - - - *Fin del caso de uso.*

Trayectoria alternativa CU30.1:

Condición: Puntos del tipo repulsivo

- CU30.1 - 1** El sistema obtiene los puntos válidos de las intersecciones del tipo repulsivo
- CU30.1 - 2** El sistema agrega los puntos del tipo repulsivo a la gráfica
- CU30.1 - 3** El sistema renderiza IU-CU30.2 Gráfica de polinomio característico con puntos de intersección del tipo repulsivo
- - - - *Fin de la trayectoria.*

Trayectoria alternativa CU30.2:

Condición: Puntos del tipo crítico

- CU30.2 - 1** El sistema obtiene los puntos válidos de las intersecciones del tipo crítico
- CU30.2 - 2** El sistema agrega los puntos del tipo crítico a la gráfica
- CU30.2 - 3** El sistema renderiza IU-CU30.3 Gráfica de polinomio característico con puntos de intersección del tipo crítico
- - - - *Fin de la trayectoria.*

Trayectoria alternativa CU30.3:

Condición: Puntos del tipo indiferente

- CU30.3 - 1** El sistema obtiene los puntos válidos de las intersecciones del tipo indiferente
- CU30.3 - 2** El sistema agrega los puntos del tipo indiferente a la gráfica
- CU30.3 - 3** El sistema renderiza IU-CU30.4 Gráfica de polinomio característico con puntos de intersección del tipo indiferente
- - - - *Fin de la trayectoria.*

4.5.31. CU31 Iniciar simulación

Descripción completa

Atributos importantes

Caso de Uso:	CU31 Iniciar simulación
Versión:	1.2
Actor:	Usuario
Propósito:	
Resumen:	El AC evoluciona un número indefinido de generaciones de acuerdo con el tiempo establecido para la velocidad
Entradas:	Ninguna
Salidas:	Ninguna
Precondiciones:	Debe existir un objeto del tipo “Cellular automata” en memoria y la simulación no debe haber iniciado
Postcondiciones:	Ninguna
Autor:	Vargas Romero Erick Efraín
Estado:	Fase de pruebas
Última modificación:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1 ⌂ Presiona **Play** vía  IU-CU31 Iniciar simulación
 - 2 ⌂ El sistema obtiene el valor de la velocidad de evolución del AC
 - 3 ⌂ El sistema inicia el timer de la simulación con el valor obtenido
 - 4 ⌂ El sistema deshabilita **Next generation**
 - 5 ⌂ El sistema actualiza el texto del botón de “Play” por “Pause”
 - 6 ⌂ El sistema cambia el ícono de **Play** por el correspondiente a **Pause**
 - 7 ⌂ El sistema calcula las siguientes generaciones del autómata según CU20 Calcular siguiente generación de acuerdo con el *valor de la velocidad*
 - 8 ⌂ El sistema muestra las siguientes generaciones calculadas del autómata según CU21 Desplegar siguiente generación
- - - *Fin del caso de uso.*

4.5.32. CU32 Detener simulación

Descripción completa

Atributos importantes

Caso de Uso:	CU32 Detener simulación
Versión:	1.2
Actor:	Usuario
Propósito:	
Resumen:	El usuario tiene la posibilidad de detener la evolución del AC, una vez hecho esto el simulador mostrará solamente la última generación calculada hasta el tiempo t
Entradas:	Ninguna
Salidas:	Ninguna
Precondiciones:	Debe existir un objeto del tipo “Cellular automata” en memoria y la simulación debe estar en ejecución

Caso de Uso:	CU32 Detener simulación
Postcondiciones:	Ninguna
Autor:	Vargas Romero Erick Efraín
Estado:	Prototipo funcional
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1  Presiona **Pause** vía la  IU-CU32 Detener simulación
 - 2  El sistema detiene el timer de la simulación
 - 3  El sistema habilita **Next generation**
 - 4  El sistema actualiza el texto del botón de “Pause” por “Play”
 - 5  El sistema cambia el ícono de **Pause** por el correspondiente a **Play**
 - 6  El sistema detiene el cálculo de las siguientes generaciones
- *Fin del caso de uso.*

4.5.33. CU33 Restablecer simulación

Descripción completa

Atributos importantes

Caso de Uso:	CU33 Restablecer simulación
Versión:	1.2
Actor:	Usuario
Propósito:	
Resumen:	El usuario tiene la posibilidad de limpiar la generación actual y regresar al estado cero del AC. El sistema mostrará el espacio de evolución con las dimensiones establecidas una configuración de células muertas, es decir, todas las células pasan al estado 0. Las gráficas se renderizan quitando todos los puntos calculados hasta el tiempo t .
Entradas:	Ninguna
Salidas:	Ninguna
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Autor:	Vargas Romero Erick Efraín
Estado:	Fase de pruebas
Última revisión:	25 de noviembre de 2020

Trayectorias del Caso de Uso

Trayectoria principal: Principal

- 1  Presiona **Clear** vía  IU-CU33 Restablecer simulación
- 2  El sistema itera célula a célula en el espacio de evoluciones estableciéndolas al estado 0

- 3 ○ El sistema actualiza el valor de la variable que almacena la generación a 0
- 4 ○ El sistema actualiza el valor de la variable que almacena la población a 0
- 5 ○ El sistema elimina los puntos calculados en la gráfica de población
- 6 ○ El sistema renderiza IU-CU33.1 Gráfica de población sin los puntos
- 7 ○ El sistema elimina los puntos calculados en la gráfica de entropía
- 8 ○ El sistema renderiza IU-CU33.2 Gráfica de entropía sin los puntos
- 9 ○ El sistema renderiza IU-CU33.3 Espacio de evolución del AC

- - - - *Fin del caso de uso.*

4.6. Reglas de negocio

RN01: Valor permitido para la dimensión del espacio

Descripción: El valor de la dimensión está compuesto por un conjunto de al menos un dígito y a lo más 5 dígitos, comprendido en el rango de 3 a 10000 y debe cumplir con la siguiente expresión regular: [0 : 9].

Tipo: Expresión regular

Nivel: Obligatorio

RN02: Cálculo de ID para una regla totalista

Descripción: El ID de la regla totalista se calcula obteniendo los valores de cada configuración de la regla, los cuales pueden ser 0 o 1, se concatenan para obtener un número binario y posteriormente se realiza la conversión al sistema decimal cumpliendo con la siguiente expresión regular: [0 : 9]. Matemáticamente se expresa de la siguiente forma:

$$ID_{bin} = \sum_{i=0}^{31} totalista[i] \quad (\text{Ec. 4.6.1})$$

$$ID = dec(ID_{bin}) \quad (\text{Ec. 4.6.2})$$

Tipo: Expresión matemática

Nivel: Obligatorio

RN03: Valor permitido para definir la densidad de aparición de las células vivas para la regla totalista

Descripción: El valor de la variable que almacena la densidad de aparición de células vivas está compuesto por al menos un dígito y un máximo de 6 dígitos, es del tipo double comprendido en un rango de 0.00000 a 1.00000 y debe cumplir con la siguiente expresión regular: [0–9]{1}(.[.])[0–9]{1,5}?. Este valor determina el número de células vivas para las configuraciones de la regla totalista.

Tipo: Expresión regular

Nivel: Obligatorio

RN04: Valores de la regla totalista

Descripción: Los valores de una regla totalista se calculan obteniendo el ID de la regla, el cual se encuentra en formato decimal, posteriormente se realiza la conversión al sistema binario y cada uno de los 32 dígitos obtenidos se asignan a las configuraciones de la regla totalista. Matemáticamente se expresa de la siguiente forma:

$$ID_{bin} = bin(ID) \quad (\text{Ec. 4.6.3})$$

$$totalista[i] = ID_{bin}[i], i = 0, \dots, 31 \quad (\text{Ec. 4.6.4})$$

Tipo: Expresión matemática

Nivel: Obligatorio

RN05: Valor permitido para las variables condición de nacimiento y supervivencia

Descripción: El valor de las variables N_{min} , N_{max} , S_{min} , S_{max} está compuesto por un dígito. Comprendido en el rango de 0 a 8 y debe cumplir con la siguiente expresión regular [0 : 9].

Tipo: Expresión regular

Nivel: Obligatorio

RN06: Valor permitido para definir la densidad de aparición de las células vivas

Descripción: El valor de la variable que almacena la densidad de aparición de células vivas está compuesto por al menos un dígito y un máximo de 6 dígitos, es del tipo double comprendido en un rango de 0.00000 a 1.00000 y debe cumplir con la siguiente expresión regular: [0 – 9]{1}([.])[0 – 9]{1,5})?.

Tipo: Expresión regular

Nivel: Obligatorio

RN07: Nombre de archivo a exportar

Descripción: El nombre del archivo a exportar se establece mediante el tipo de regla. Si la regla es de tipo totalista, se obtiene el ID mediante el uso de RN02 ID de una regla totalista y se concatena al prefijo “CA-” y al sufijo “.dat”. En cambio, si la regla es de tipo semitotalista, se obtiene $S_{min}, S_{max}, N_{min}, N_{max}$ y se concatena al prefijo “CA-S”, seguido de los valores de supervivencia, “B”, seguido de los valores de nacimiento y el sufijo “.ST.dat”.

Para el caso de la regla totalista, debe cumplir con la siguiente expresión regular: CA – [0 – 9]{0,10} – T[.]dat. Para el caso de la regla semitotalista, debe cumplir con la siguiente expresión regular: CA – S[0 – 9]{2} – B[0 – 9]{2} – ST[.]dat

Tipo: Expresión regular

Nivel: Obligatorio

RN08: Valor permitido para tamaño de la célula

Descripción: El valor del tamaño de la célula está compuesto por al menos un dígito y un máximo de 2 dígitos, comprendido en el rango de 1 a 20 y debe cumplir con la siguiente expresión regular: [0 : 9]{1,2}.

Tipo: Expresión regular

Nivel: Obligatorio

RN09: Valor permitido para definir velocidad de evolución del AC

Descripción: El valor de la variable está compuesto por al menos un dígito y un máximo de 4 dígitos, comprendido en el rango de 0 a 5000 y debe cumplir con la siguiente expresión regular: [0 : 9]{1,4}.

Tipo: Expresión regular

Nivel: Obligatorio

RN10: Valor permitido para el escalar de proyección

Descripción: El valor del escalar está compuesto por un conjunto de al menos un dígito y a lo más 3 dígitos, comprendido en el rango de 0 a 100 y debe cumplir con la siguiente expresión regular [0 : 9]{1,3}.

Tipo: Expresión regular

Nivel: Obligatorio

RN11: Color de células con gradiente

Descripción: Cada célula tiene asignado un valor numérico (positivo, negativo o cero). Si el valor de la célula es positivo y además se encuentra en el rango de 1 a 100000, se realizan las siguientes operaciones para obtener el color de gradiente correspondiente, donde $rango = 12500$:

$$offset = rango(color_{células_vivas} + color_{célula_actual} > 0x00FFFFFF) \quad (\text{Ec. 4.6.5})$$

Finalmente, el color de la célula con gradiente está dado por la siguiente ecuación:

$$color_{célula_actual_con_gradiente} = color_{célula_actual} + offset(8) \quad (\text{Ec. 4.6.6})$$

Tipo: Expresión matemática

Nivel: Obligatorio

RN12: Valor permitido para el color de una célula o rejilla

Descripción: El valor del color está compuesto por al menos 1 dígito y un máximo 3 dígitos para el rojo, azul y verde, respectivamente. En el caso de ser valor hexadecimal, el valor del color está compuesto por el carácter # como prefijo, además de al menos 3 caracteres y un máximo 6 caracteres comprendidos en el rango A-F o dígitos. El valor del color RGB debe cumplir con la siguiente expresión regular: [0 – 9]1,3[0 – 9]1,3[0 – 9]1,3. El valor del color hexadecimal debe cumplir con la siguiente expresión regular: #[a – fA – F0 – 9]3,6.

Tipo: Expresión regular

Nivel: Obligatorio

RN13: Posición de una célula

Descripción: Dada la coordenada del evento click y el tamaño de una célula se obtiene la posición en el espacio de evoluciones de la célula $c_{i,j}$ realizando la siguiente operación:

$$(x_c, y_c) = \left(\frac{x}{cellSize}, \frac{y}{cellSize} \right)$$

Tipo: Expresión matemática

Nivel: Obligatorio

RN14: Evaluación del espacio de evolución

Descripción: Se evalúa el espacio de evoluciones hasta la célula $c_{i,j}$ donde $i, j \in \mathbb{Z}$ esta evaluación se realiza evaluando una submatriz de dimensión 3x3 para el caso en que el tipo de vecindad de evaluación sea del tipo *Moore* es decir se itera desde $[-1, 1]$ y se suma este valor a la posición i y j (esto se realiza de forma anidada es decir dentro de dos ciclos que iteran en el rango descrito anteriormente) si la vecindad es del tipo *vonNeumann* se realiza la evaluación de las células superior, inferior, izquierda y derecha. Si el tipo de espacio es cerrado es decir tiene forma de *toroide* se toma la posición actual en la que se está evaluando la submatriz y se aplica álgebra modular para obtener un índice válido, es decir:

$$(i, j) = \begin{cases} (i + n) \% n & \text{si } i < 0 \\ (j + n) \% n & \text{si } j < 0 \\ i \% n & \text{si } i \geq n \\ j \% n & \text{si } j \geq n \\ (i, j) & \text{otro caso} \end{cases}$$

Por otra parte si el tipo de espacio es abierto entonces se debe de evaluar si los índices i y j son válidos con la siguiente expresión

$$(i, j) = \begin{cases} 0 & \text{si } i < 0 \\ 0 & \text{si } j < 0 \\ 0 & \text{si } i \geq n \\ 0 & \text{si } j \geq n \\ (i, j) & \text{otro caso} \end{cases}$$

Tipo: Expresión matemática

Nivel: Obligatorio

RN15: Evaluación de la célula totalista

Descripción: El sistema obtiene las 32 configuraciones de la regla totalista y sus valores. Si el valor de la célula $c_{i,j}$ y de sus 4 vecinos ($c_{i-1,j}, c_{i+1,j}, c_{i,j-1}, c_{i,j+1}$) corresponde a alguna de las configuraciones, actualiza el valor para la siguiente generación con el valor definido para esa configuración.

Tipo: Expresión matemática

Nivel: Obligatorio

RN16: Valor de célula

Descripción: Cada célula $c_{i,j}$ del espacio de evoluciones cumple $c_{i,j} = \{-1, 0, 1\}$ siendo así se colorea cada célula de acuerdo a lo siguiente:

$$c_{i,j} = \begin{cases} c_{i,j} = 1 & \text{entonces color de una célula viva} \\ c_{i,j} = -1 & \text{entonces color de una célula de proyección} \\ c_{i,j} = 0 & \text{entonces color de una célula muerta} \end{cases}$$

Tipo: Expresión matemática

Nivel: Obligatorio

RN17: Contabilización de vecinos alrededor de $c_{i,j}$

Descripción: Existe un contador inicializado en cero. Dependiendo del tipo de vecindad (Moore o von Neumann) se calcularán los vecinos de la célula $c_{i,j}$ cuyo valor sea el de una célula viva (1) o de proyección (-1) de la forma siguiente:

- Moore: $\text{contador} = c_{i-1,j+1} + c_{i-1,j} + c_{i-1,j-1} + c_{i,j+1} + c_{i,j-1} + c_{i+1,j+1} + c_{i+1,j} + c_{i+1,j-1} + c_{i,j}$
- von Neumann: $\text{contador} = c_{i-1,j} + c_{i+1,j} + c_{i,j-1} + c_{i,j+1} + c_{i,j}$

Tipo: Expresión matemática

Nivel: Obligatorio

RN18: Evaluación de la célula semitotalista

Descripción: El sistema obtiene las reglas de nacimiento y supervivencia y de acuerdo con la RN17Contabilización de vecinos, el valor de la célula para la siguiente generación será calculado de la siguiente forma:

$$c_{i,j} = \begin{cases} 1 & \text{si } \begin{cases} c_{i,j} = 1 & \text{y } S_{min} \leq \text{contador} \leq S_{max} \\ c_{i,j} = 0 & \text{y } B_{min} \leq \text{contador} \leq B_{max} \\ \text{otro caso} & \end{cases} \\ 0 & \end{cases}$$

Tipo: Expresión matemática

Nivel: Obligatorio

RN19: Cálculo de la densidad de población

Descripción: Se realiza el cálculo de la densidad de acuerdo a la siguiente expresión matemática:

$$x = \frac{\text{células vivas}}{\text{número total de células}}$$

Tipo: Expresión matemática

Nivel: Obligatorio

RN20: Cálculo de la entropía

Descripción: Se realiza el cálculo de la entropía de acuerdo a la siguiente expresión matemática:

$$\text{entropía} = -(x \cdot \log_2(x) + (1-x) \cdot \log_2(1-x))$$

Tipo: Expresión matemática

Nivel: Obligatorio

RN21: Cálculo del polinomio característico

Descripción: Se realiza el cálculo del polinomio con la siguiente expresión matemática:

$$p_{t+1} = \sum_{v=S_{min}}^{S_{max}} \binom{n-1}{v} p_t^{v+1} q_t^{n-v-1} + \sum_{v=N_{min}}^{N_{max}} \binom{n-1}{v} p_t^v q_t^{n-v}$$

n es el valor de células de la vecindad 9 para Moore y 5 para von Neumann

Tipo: Expresión matemática

Nivel: Obligatorio

RN22: Punto candidato

Descripción: Un punto candidato es el resultado de evaluar algún x en la función $f(x)$, en donde $f(x)$ es el polinomio característico y que además cumple que al evaluar el mismo x en $g(x)$, en donde $g(x)$ es la recta identidad, el resultado de ambas evaluaciones es idéntico con una precisión de $1x10^{-6}$ decimales.

Tipo: Expresión matemática

Nivel: Obligatorio

RN23: Punto válido para la gráfica

Descripción: Un punto válido para la gráfica es el promedio de todos los puntos del conjunto de X . Es decir, $X = \frac{1}{n} \sum_{i=1}^n x_i = \frac{x_1 + x_2 + \dots + x_n}{n}$ y que además, la diferencia entre cada valor consecutivo del conjunto de $\{X\}$ es a lo más $1x10^{-6}$.

Tipo: Expresión matemática

Nivel: Obligatorio

RN24: Tipo de punto fijo

Descripción: Existen diferentes tipos de punto fijo, denotados por x^* y están definidos de la siguiente forma:

- Punto atractivo: si $|f'(x^*)| \pm \epsilon < 1$
- Punto repulsivo: si $|f'(x^*)| \pm \epsilon > 1$
- Punto indiferente: si $|f'(x^*)| \pm \epsilon = 1$
- Punto crítico: si $f'(x^*) \pm \epsilon = 0$

donde ϵ está definido como $1x10^{-6}$

Tipo: Expresión matemática

Nivel: Obligatorio

RN25: Puntos pertenecientes a la recta identidad

Descripción: Cada punto (x, x) de la recta identidad cumple $I = [0, 1]$ además $|x_i - x_{i+1}| \leq \epsilon$, donde $\epsilon = 1x10^{-6}$

Tipo: Expresión matemática

Nivel: Obligatorio

4.7. Catálogo de mensajes

Mensaje: MSG-CU08

Your AC has been saved successfully.

Mensaje: MSG-CU08.1

Please select a path first.

5

Fase 2: Interfaz del usuario

5.1. Prototipos

Una vez analizado el sistema y las partes que lo componen, se realiza el diseño de los prototipos, de esta forma es más fácil ubicar las posiciones de los componentes del simulador, así como organizarlos y que los desarrolladores logren ver e interpretar de una forma más fácil lo que se quiere lograr. Hasta el momento se han definido cuatro prototipos con los requerimientos obtenidos para el simulador y se muestran a continuación.

5.1.1. Prototipo 01

Descripción: Este prototipo tiene como objetivo realizar la evaluación de un AC en dos dimensiones con las vecindades del tipo Moore o von Neumann, permitiendo especificar las reglas de nacimiento y supervivencia. Para que sea posible mostrar el autómata y sus evoluciones se han agregado los botones **Draw initial condition** para mostrar la configuración inicial, **Next generation** para que el AC calcule la siguiente generación y la muestre, **Play** para que el AC calcule un número indefinido de generaciones y las muestre, **Pause** para detener la evolución del AC, **Stop** para detener el simulador, **Clear** para limpiar el espacio de evolución, es decir, poner todas las células del espacio en estado 0

Alcance:

- Definir el tipo de vecindad
- Definir las reglas de nacimiento y de supervivencia
- Mostrar el espacio de evolución del AC
- Calcular la siguiente generación del AC
- Detener la simulación
- Limpiar el espacio de evolución
- Modificar el estado de las células en el espacio de evolución (0 o 1)

Casos de uso involucrados:

- CU02 Definir tipo de vecindad
- CU05 Definir las reglas de evaluación del AC
- CU14 Calcular siguiente generación

- CU15 Desplegar siguiente generación
- CU19 Iniciar simulación
- CU20 Detener simulación
- CU21 Restablecer simulación
- CU23 Desplegar el AC con la configuración inicial
- CU27 Desplegar ventana de configuración

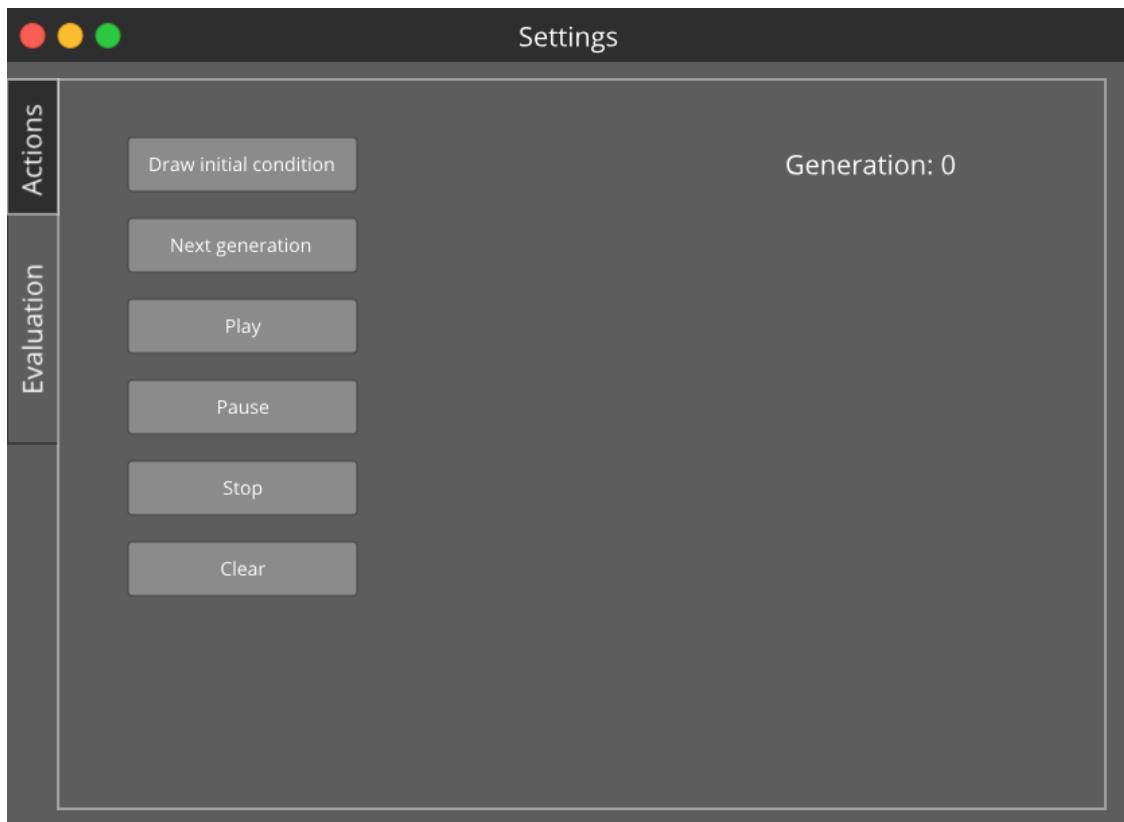
Diseño:

Fig. 5.1.1: PR01.1 Ventana de configuración - Acciones

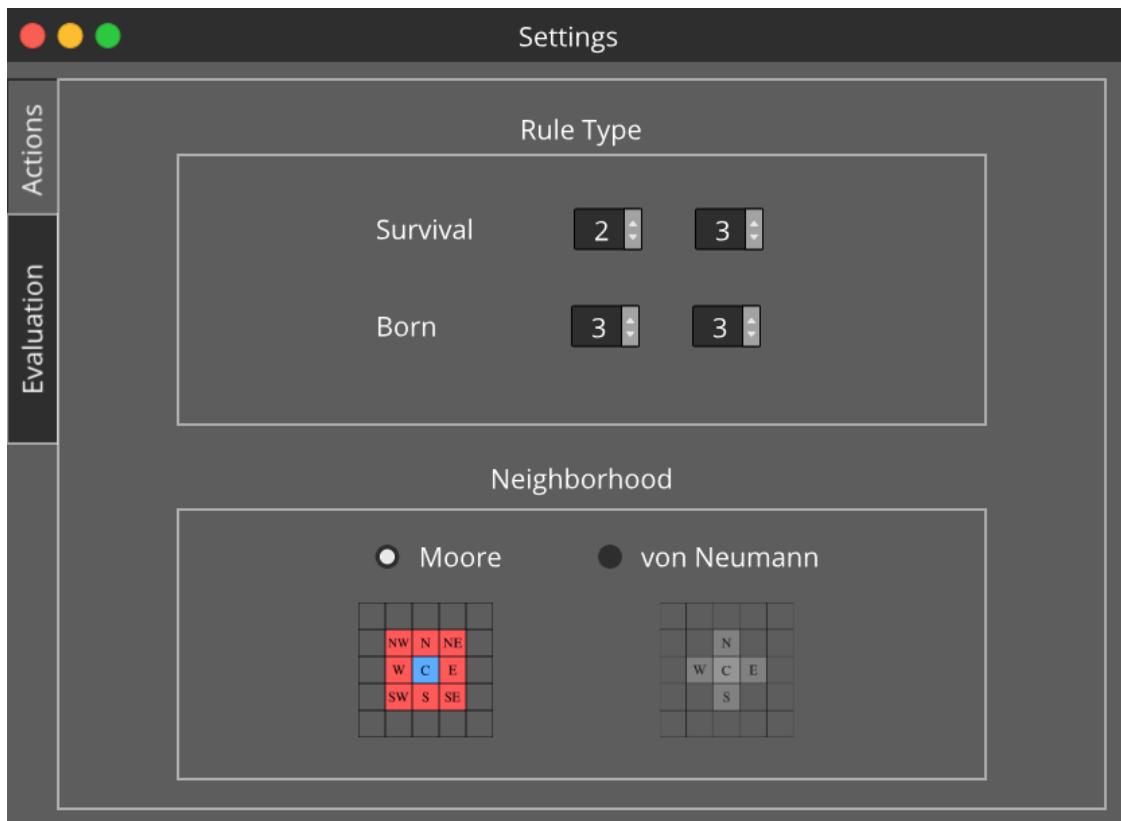


Fig. 5.1.2: PR01.2 Ventana de configuración - Evaluación

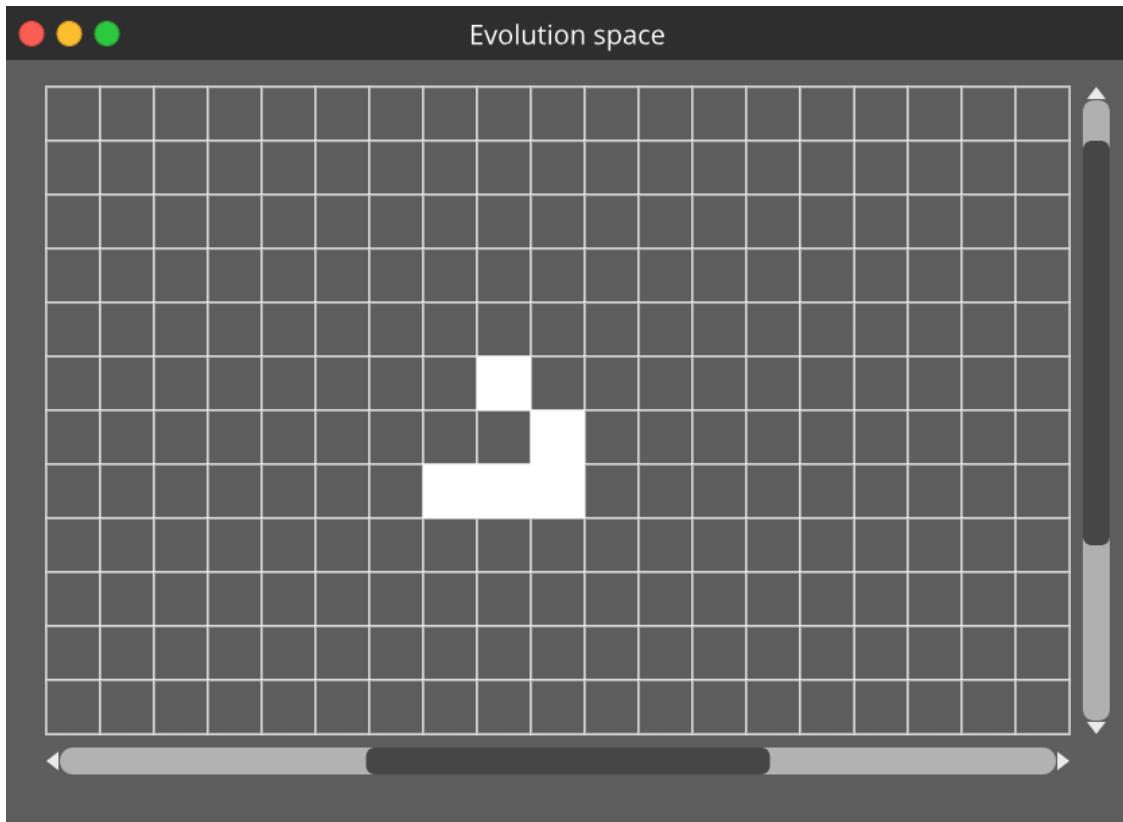


Fig. 5.1.3: PR01.3 Espacio de evolución

5.1.2. Prototipo 02

Descripción: La finalidad del prototipo es permitir modificar la configuración inicial: la anchura, altura del espacio de evoluciones, el tipo de espacio de evolución ya sea abierto o cerrado y él como será inicializado el AC. Para que lo antes descrito sea posible se han añadido los siguientes componentes: Open space el cual cambia el estado del espacio a abierto, es decir, el espacio se comporta como si fuese una matriz común, mientras que Closed space modifica el estado del espacio de evoluciones a cerrado y este se comporta como si fuese un toroide. También es posible elegir como inicializar el AC, por una parte si se presiona Empty space cada célula $c_{i,j}$ del espacio de evoluciones es igual con cero, si se presiona Random, se ingresa la densidad de células vivas (aleatoriamente) para la generación cero y finalmente si se presiona From file y posteriormente Choose es posible cargar un archivo del tipo binario que contiene un objeto con el que será inicializado el AC.

Alcance:

- Definir el tamaño del espacio
- Establecer configuración inicial del AC

Casos de uso involucrados:

- CU01 Definir tamaño del espacio
- CU07 Establecer configuración inicial del AC

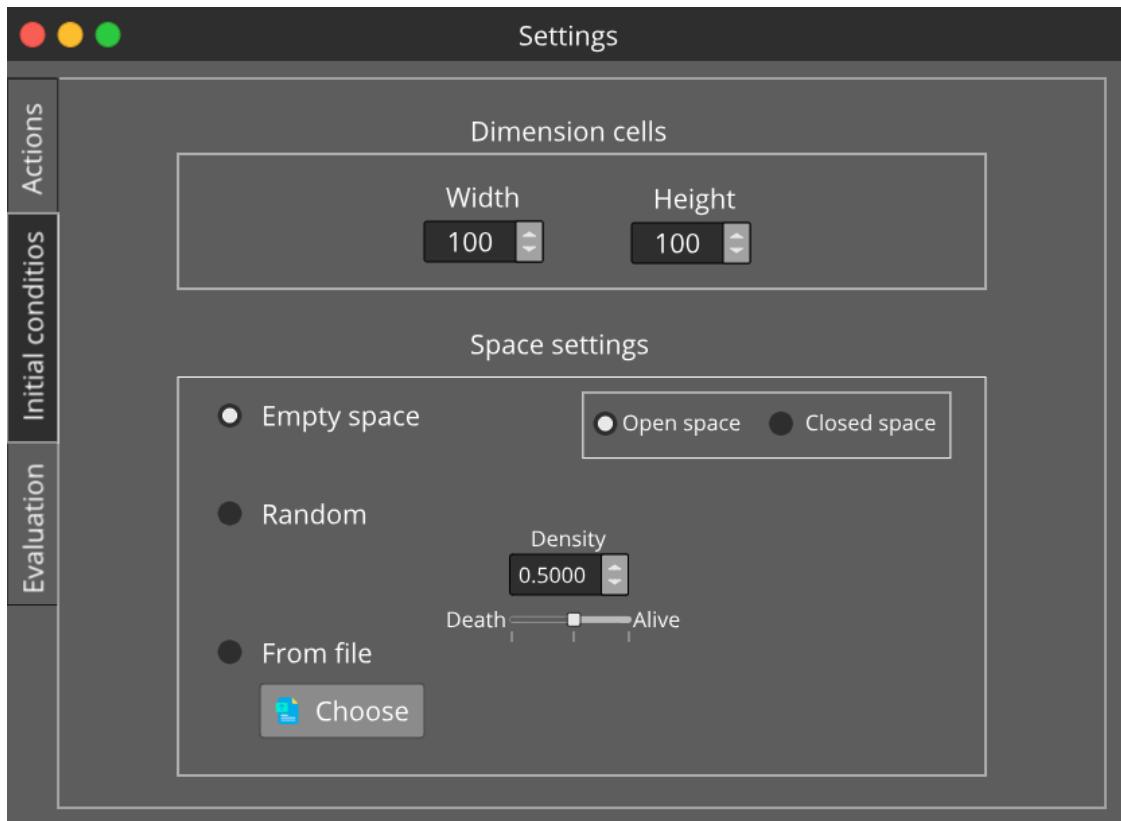
Diseño:

Fig. 5.1.4: PR02.2 Ventana de configuración - Condiciones iniciales

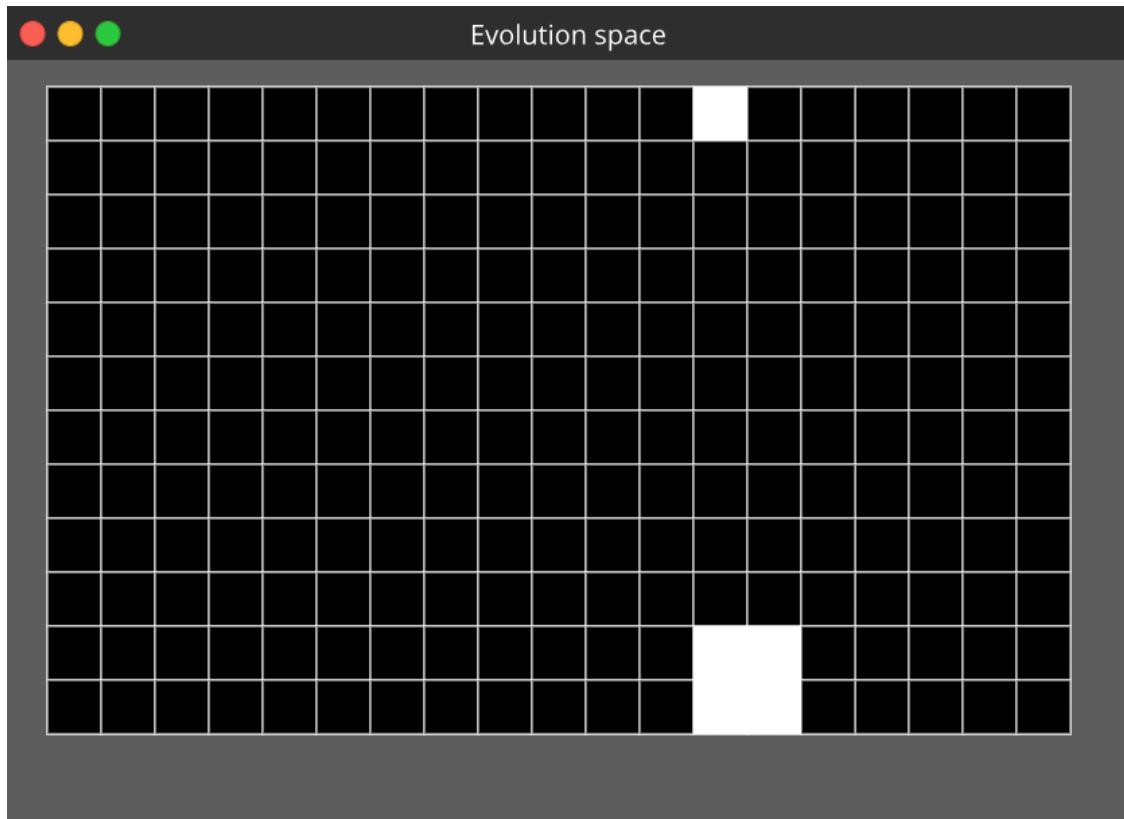


Fig. 5.1.5: PR02.2 Espacio abierto

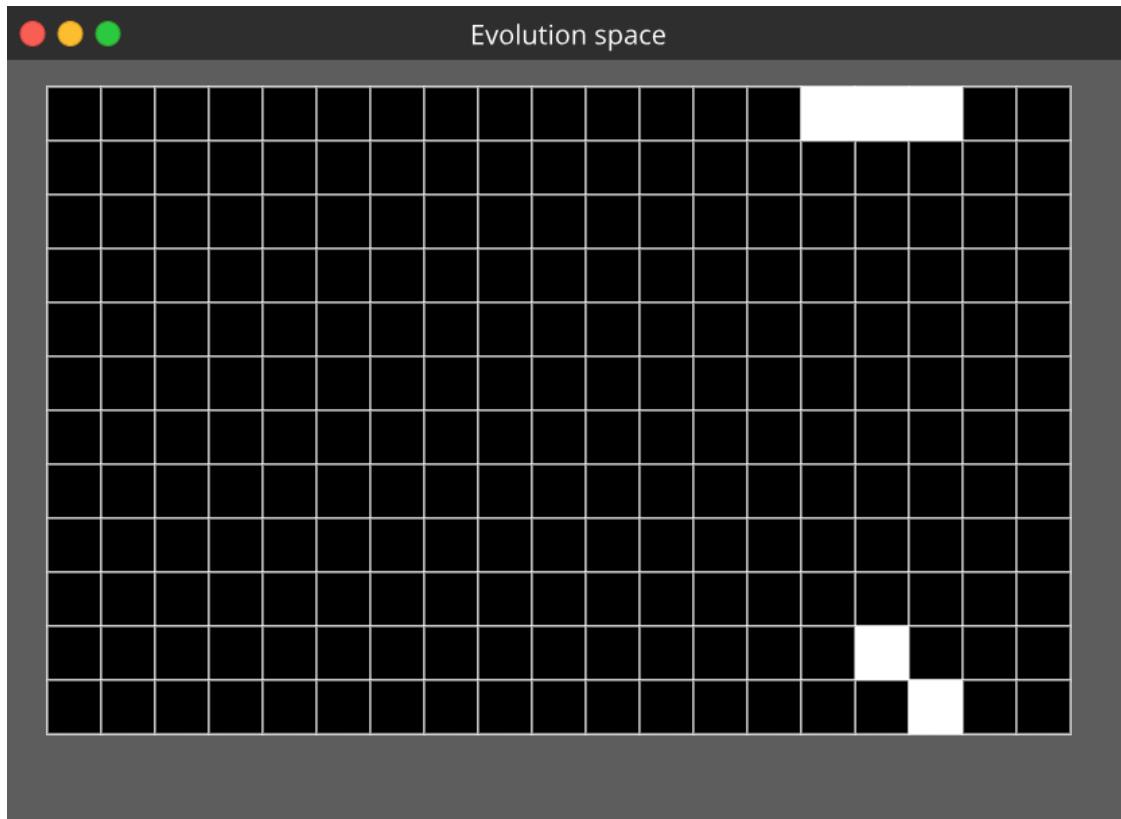


Fig. 5.1.6: PR02.3 Espacio cerrado

5.1.3. Prototipo 03

Descripción: El tercer prototipo en la pestaña *Actions* se despliega el promedio, la cantidad de células vivas y la densidad de estas mientras que en la pestaña *Space displaying* es posible modificar la velocidad con la que evoluciona el AC, habilitar la rejilla, modificar el tamaño de las células y finalmente modificar el color tanto de las células vivas, células muertas y el color de la rejilla. Para esto se han añadido los siguientes componentes, primeramente en la pestaña *Actions* hay 4 elementos gráficos del tipo *Display* en el primero se muestra la generación actual del AC, en el segundo el promedio de células vivas en la generación actual, en el tercero la cantidad de células vivas actuales y finalmente la densidad de células vivas. Por otra parte en la pestaña *Space displaying* se puede ingresar un valor numérico que permitirá cambiar velocidad con la que será calculada la siguiente generación del AC en millisegundos, el siguiente componente es similar al anterior, permite ingresar un valor numérico que permitirá escalar las células del espacio de evolución. Si *Grid* tiene el estado de verdadero la rejilla del espacio de evolución es visible en caso contrario, se oculta. Si el usuario presiona *Living cells* se elige el color con el que serán mostradas las células vivas en el espacio de evolución, si el usuario presiona *Death cells* se elige el color de las células muertas y finalmente *Grid lines* cambia el color de la rejilla.

Alcance:

- Definir el tamaño de las células
- Definir el color de las células vivas y muertas
- Establecer el estado de la rejilla
- Definir el color de la rejilla

- Definir la velocidad de evolución del AC

Casos de uso involucrados:

- CU03 Definir el tamaño de las células
- CU04 Definir el color de las células
- CU06 Definir la velocidad de evolución del AC
- CU09 Establecer el estado de la rejilla

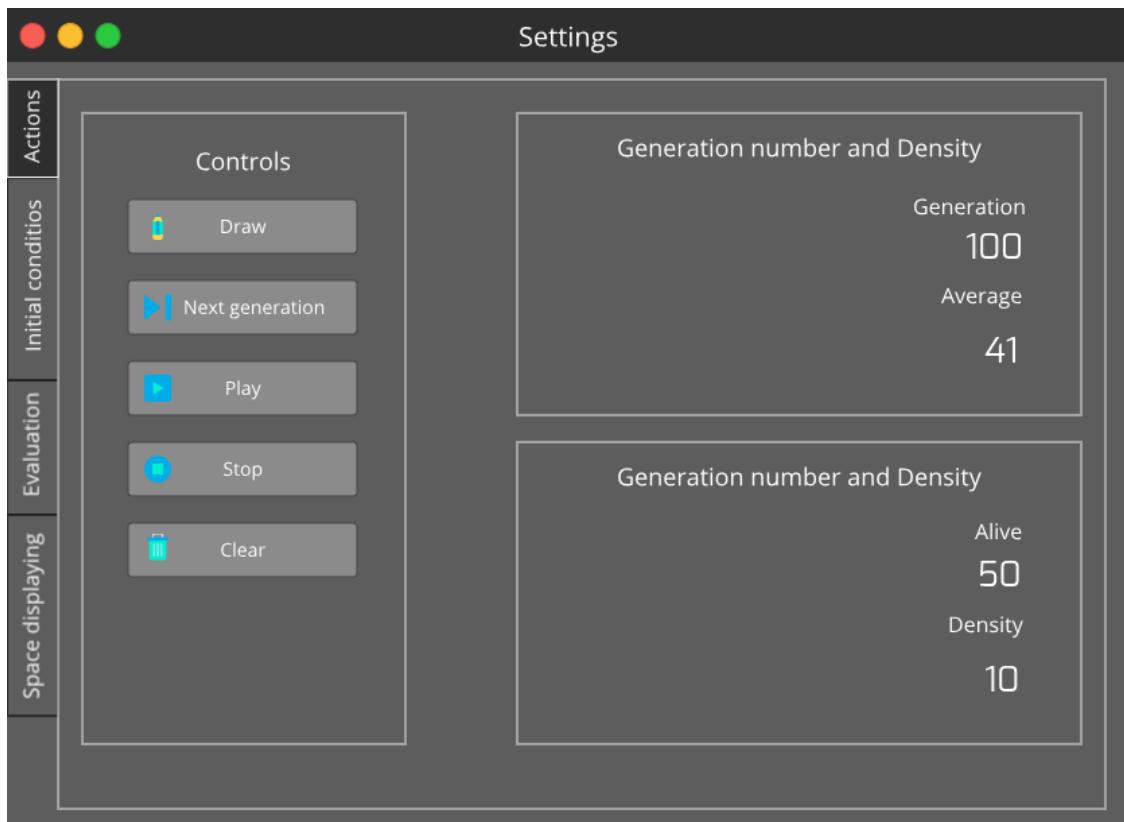
Diseño:

Fig. 5.1.7: PR03.1 Ventana de configuración - Condiciones iniciales

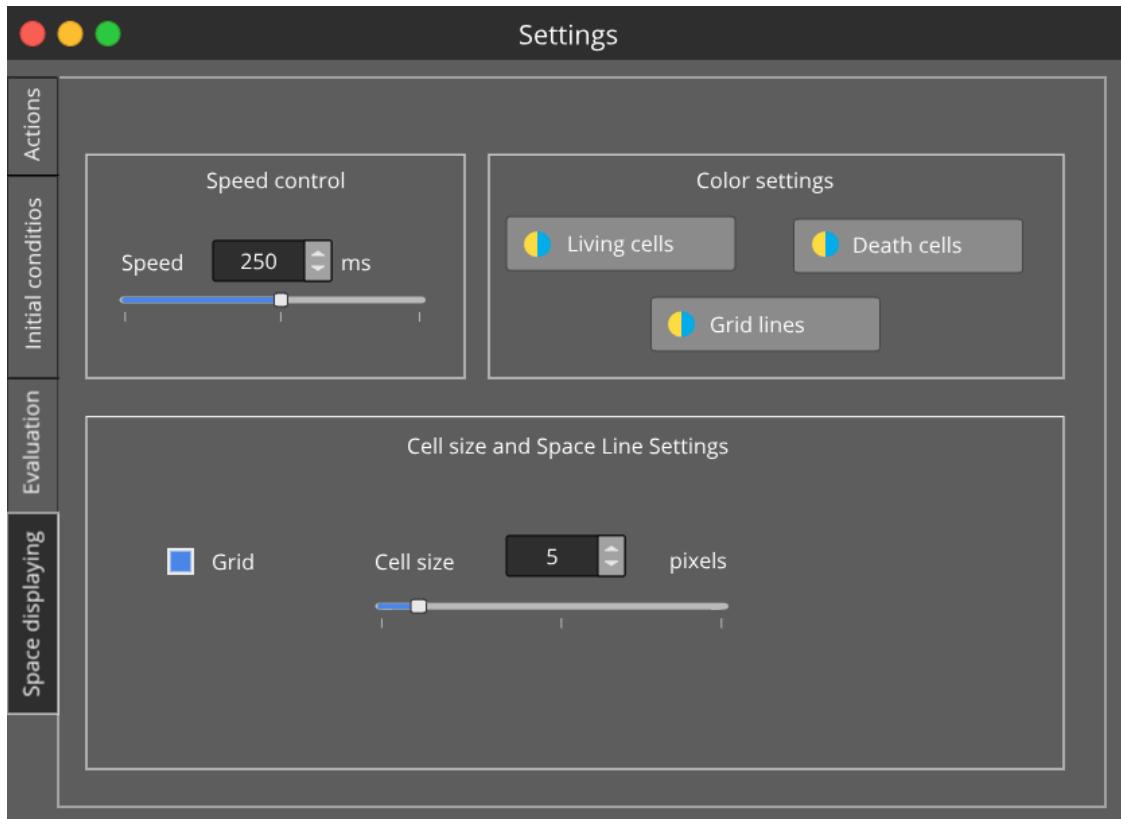


Fig. 5.1.8: PR03.2 Ventana de configuración - Espacio abierto

5.1.4. Prototipo 04

Descripción: En el cuarto prototipo en la pestaña *Actions* se han añadido más elementos, el primero permite elegir una ruta en la cual se almacenará un archivo binario que contiene un objeto del tipo "Cellular automata", el segundo exporta dicho objeto a la ruta que fue seleccionada previamente. Adicionalmente en la pestaña *Graphics* se realiza el cálculo del polinomio característico para la regla especificada, se realiza la gráfica de la población y entropía en la generación actual y finalmente es posible visualizar la siguiente generación del AC realizando una proyección en tres dimensiones. Por otra parte en la pestaña *Space displaying* se puede modificar el valor de un escalar de proyección y finalmente es posible cambiar el color de las células de proyección. Para esto se han añadido para la pestaña *Actions* dos botones **Choose path** que abre una ventana en la que se muestra el árbol de directorios del sistema y permite al usuario seleccionar una ruta en la que se almacenará un archivo binario cuyo contenido es un objeto del tipo "Cellular automata", si se presiona **Export space** este archivo se genera y es almacenado en la ruta previamente seleccionada. En la pestaña *Graphics* se visualizan tres gráficas, la primera muestra el polinomio característico de la regla actual, la recta identidad y los puntos de intersección, la segunda gráfica muestra la población hasta la generación actual y finalmente una gráfica que muestra la entropía hasta la generación actual, estas dos últimas añaden un punto a su respectiva gráfica cada que se calcula una nueva generación. En *Space displaying* se ingresa un valor numérico el cual será el valor del escalar de proyección, este permite obtener una submatriz que evoluciona de forma normal y el resto del espacio de evoluciones se mantiene estático y se desplaza con dirección nor-este y al presionar **Projection cells** se selecciona el color que tendrán las células de proyección en el espacio de evoluciones.

Alcance:

- Definir el color de las células e proyección

- Exportar el objeto "Cellular automata"
- Calcular polinomio característico
- Graficar polinomio característico
- Calcular entropía
- Graficar entropía
- Obtener población actual
- Graficar población
- Calcular puntos de intersección
- Graficar puntos de intersección
- Graficar la recta identidad

Casos de uso involucrados:

- CU04 Definir el color de las células
- CU08 Exportar el objeto "Cellular automata"
- CU10 Calcular polinomio característico
- CU11 Graficar polinomio característico
- CU12 Calcular entropía
- CU17 Graficar población
- CU15 Desplegar siguiente generación
- CU16 Definir escalar de proyección
- CU18 Graficar entropía
- CU24 Graficar recta identidad
- CU25 Calcular puntos de intersección
- CU26 Graficar puntos de intersección

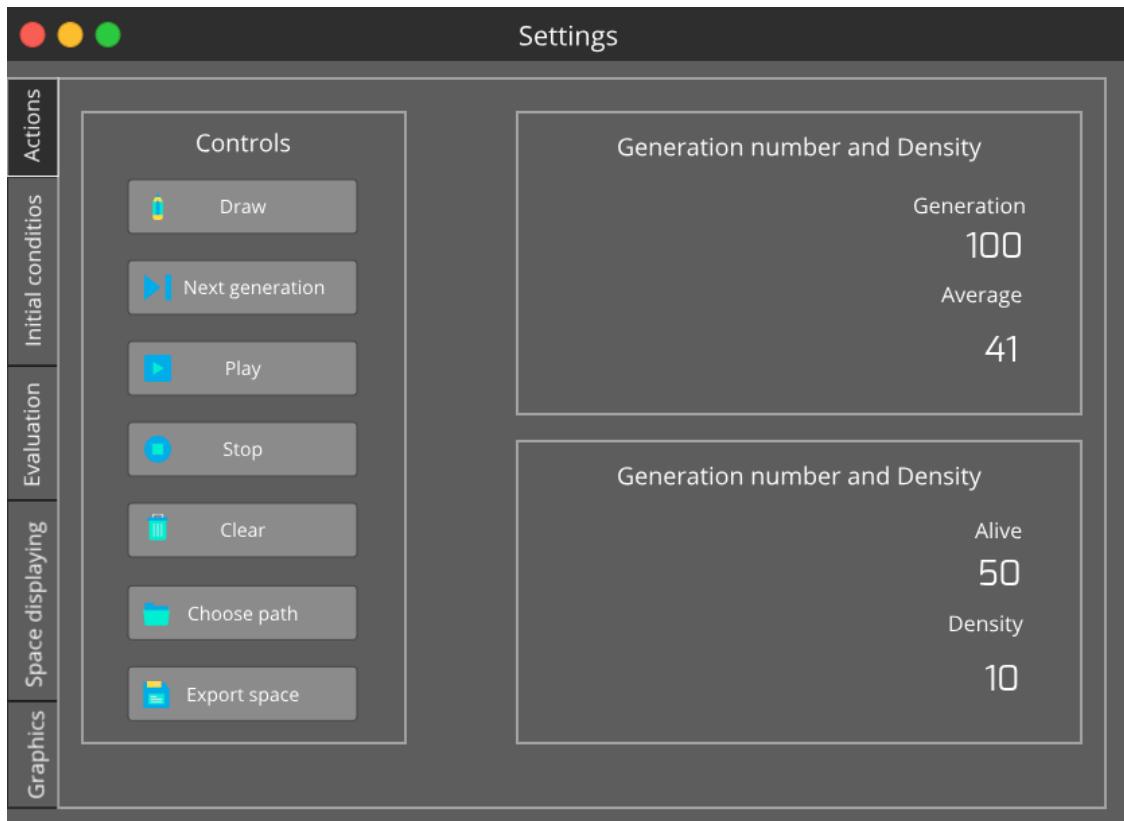
Diseño:

Fig. 5.1.9: PR04.1 Ventana de configuración - Acciones

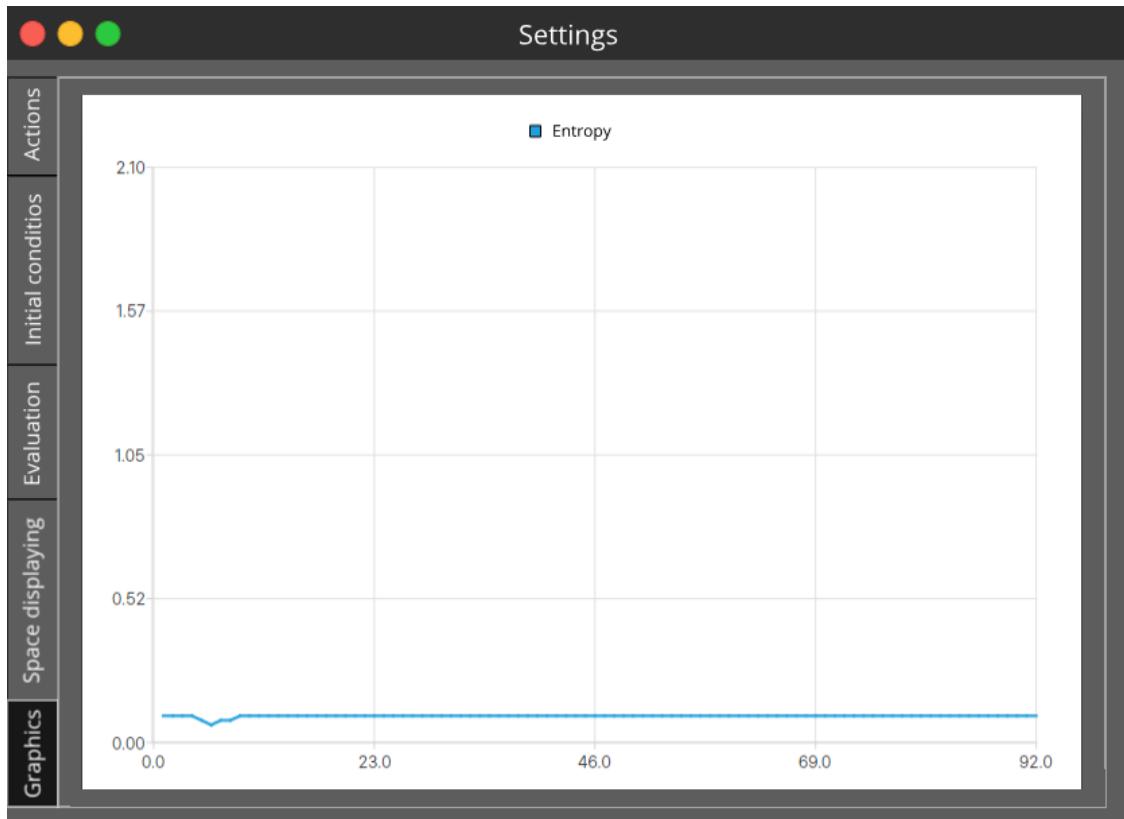


Fig. 5.1.10: PR04.2 Ventana de configuración - Entropía

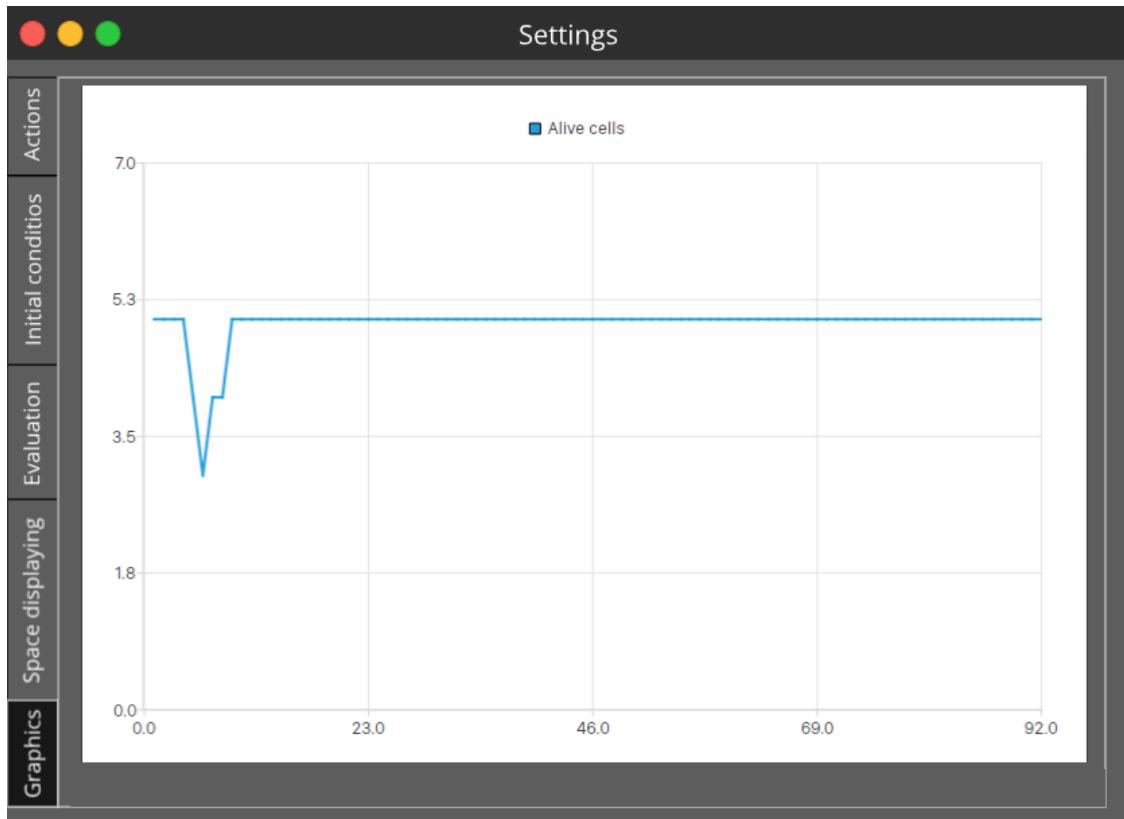


Fig. 5.1.11: PR04.3 Ventana de configuración - Población

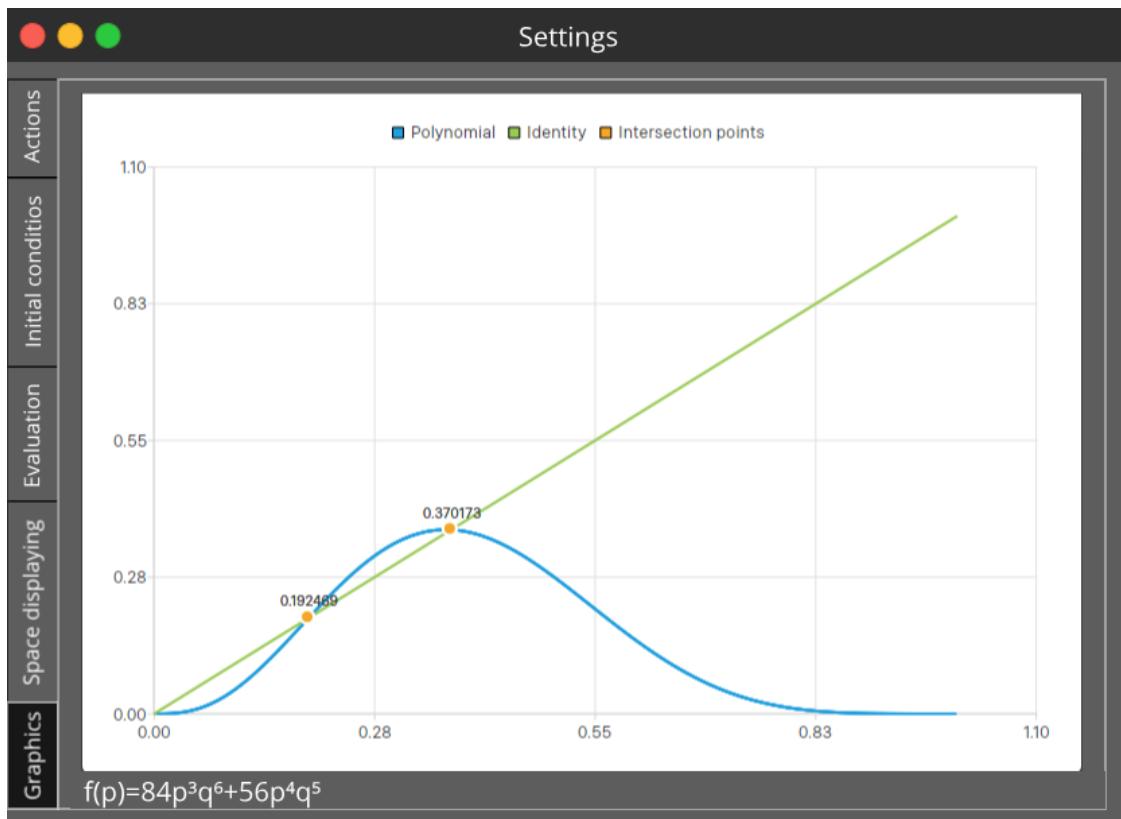


Fig. 5.1.12: PR04.4 Ventana de configuración - Polinomio

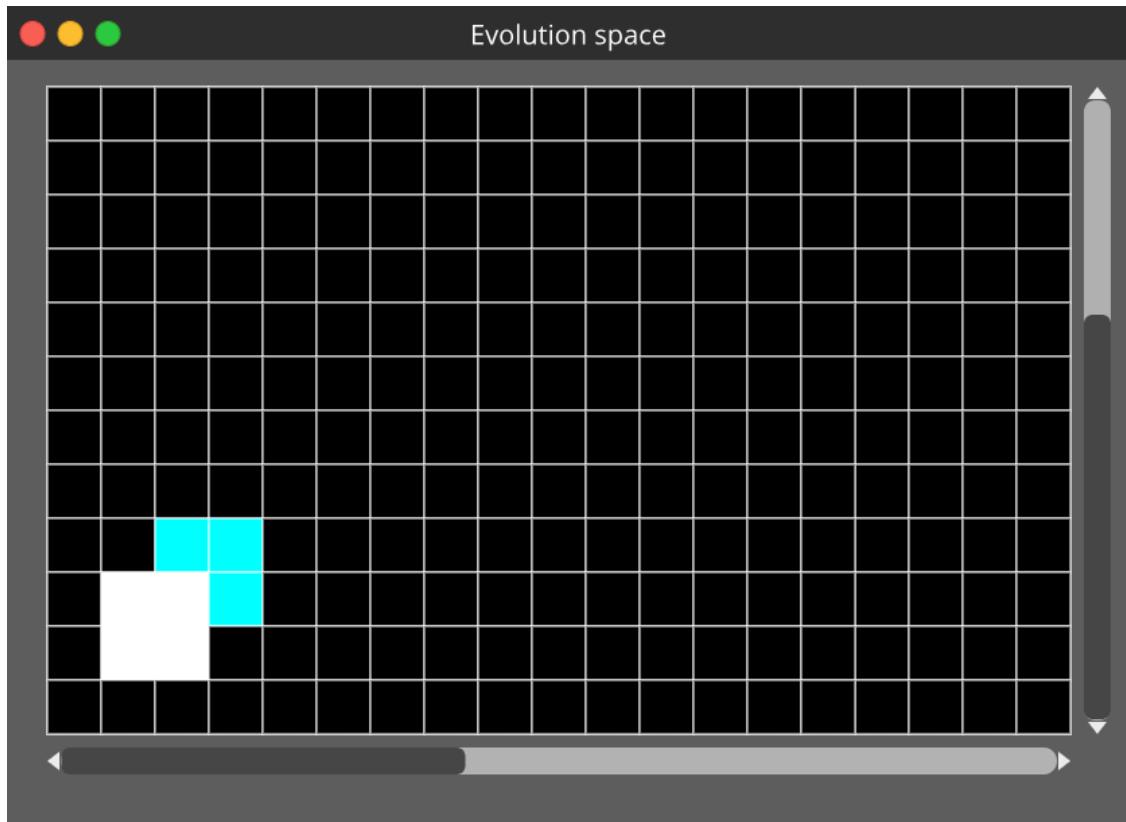


Fig. 5.1.13: PR04.5 Proyección

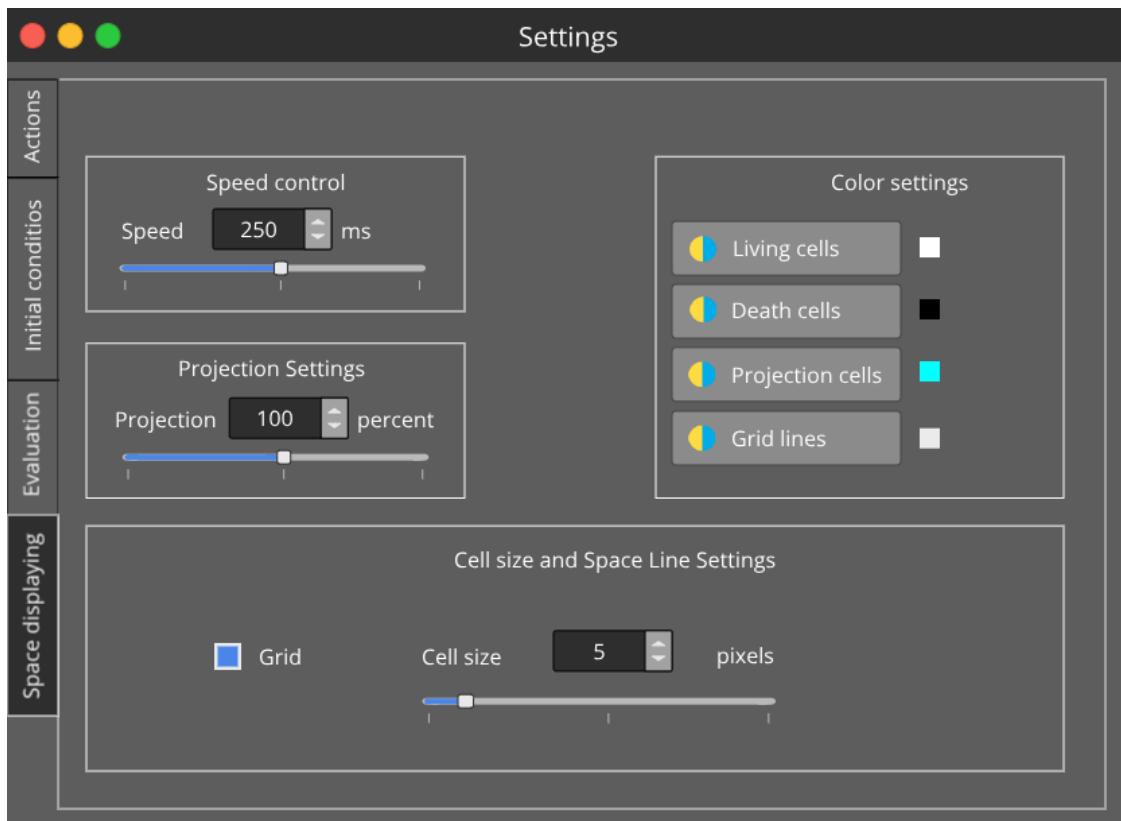


Fig. 5.1.14: PR04.6 Ventana de configuración - Visualización del espacio

6

Fase 3: Desarrollo

En esta fase se realiza el desarrollo de los prototipos en el lenguaje de programación previamente determinado, en este caso se utilizó *C++* con el framework *Qt* con la finalidad de hacer un simulador multiplataforma que se pueda ejecutar en sistemas Windows, Linux y Mac.

A continuación se muestra el avance de los prototipos descritos en la fase anterior.

6.1. Pantallas

6.1.1. IU-CU01: Definir la altura del espacio

Objetivo: El usuario podrá modificar el número de células que conforman la altura del espacio de evoluciones

Diseño: Esta pantalla aparece al hacer click en la pestaña **Initial conditions** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Altura

Salidas: Altura del espacio de evoluciones del AC

Comandos:

- \wedge : Aumenta en una unidad el tamaño de la altura
- \vee : Disminuye en una unidad el tamaño de la altura

Mensajes: Ninguno

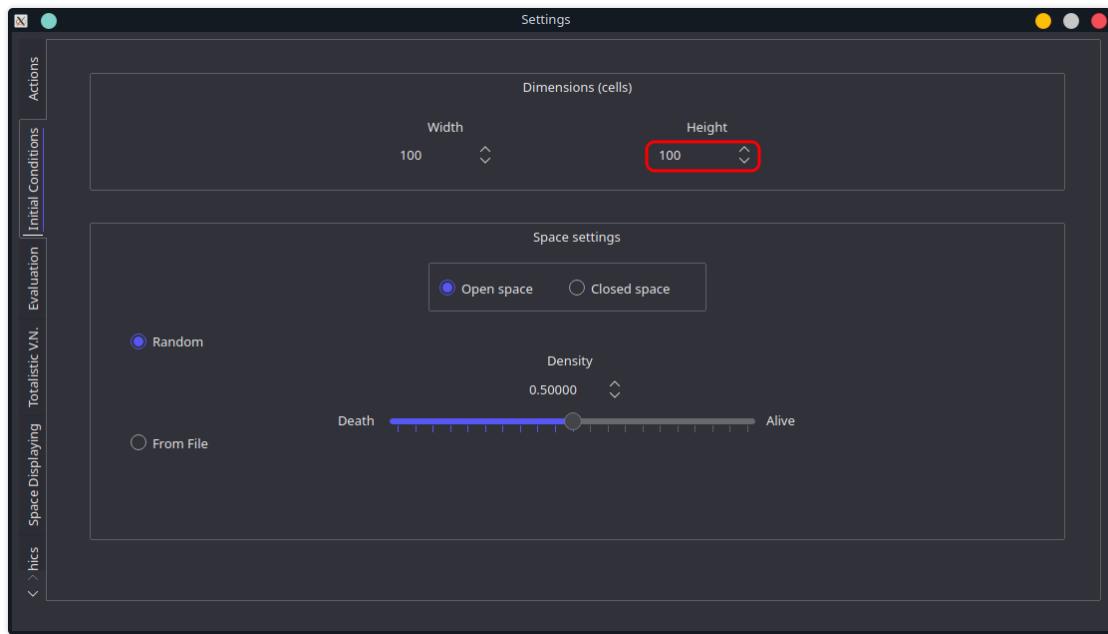


Fig. 6.1.1: IU-CU01 Definir la altura del espacio

6.1.2. IU-CU01.1: Definir la anchura del espacio

Objetivo: El usuario podrá modificar el número de células que conforman la anchura del espacio de evoluciones

Diseño: Esta pantalla aparece al hacer click en la pestaña **Initial conditions** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: anchura

Salidas: anchura del espacio de evoluciones del AC

Comandos:

- ^ : Aumenta en una unidad el tamaño de la anchura
- v : Disminuye en una unidad el tamaño de la anchura

Mensajes: Ninguno

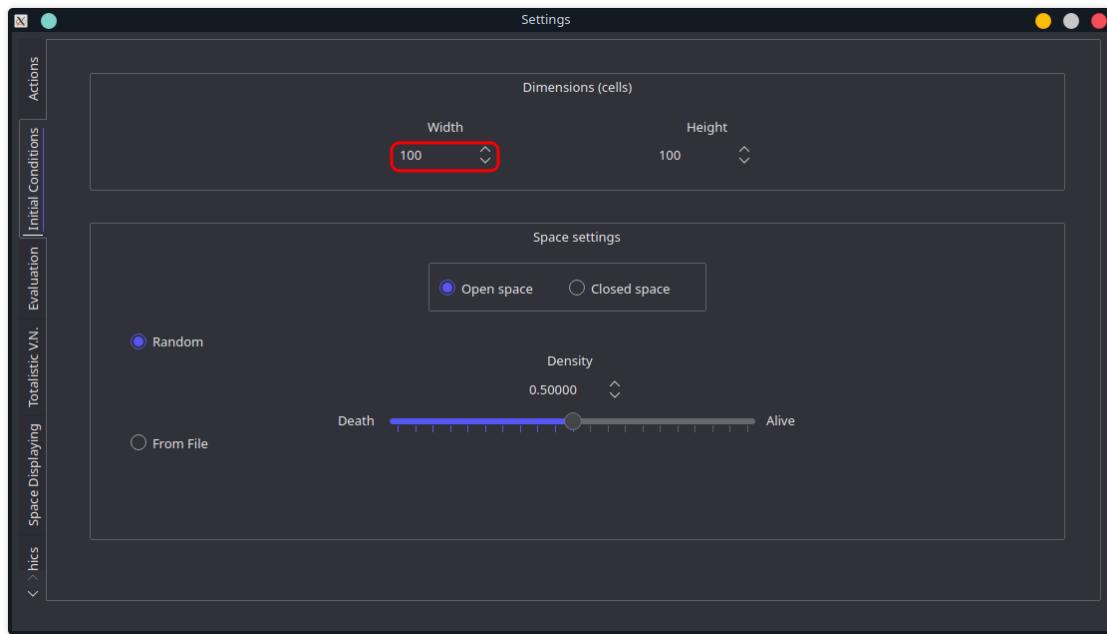


Fig. 6.1.2: IU-CU01.1 Definir la anchura del espacio

6.1.3. IU-CU02: Definir el tipo de vecindad como Moore

Objetivo: El usuario define el tipo de vecindad con la que se realizará la evaluación del AC como tipo Moore

Diseño: Esta pantalla aparece al hacer click en la pestaña **Evaluation** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Ninguna

Salidas: Ninguna

Comandos:

- Moore : Define *Moore* como el tipo de vecindad con la que se evalúa el AC

Mensajes: Ninguno

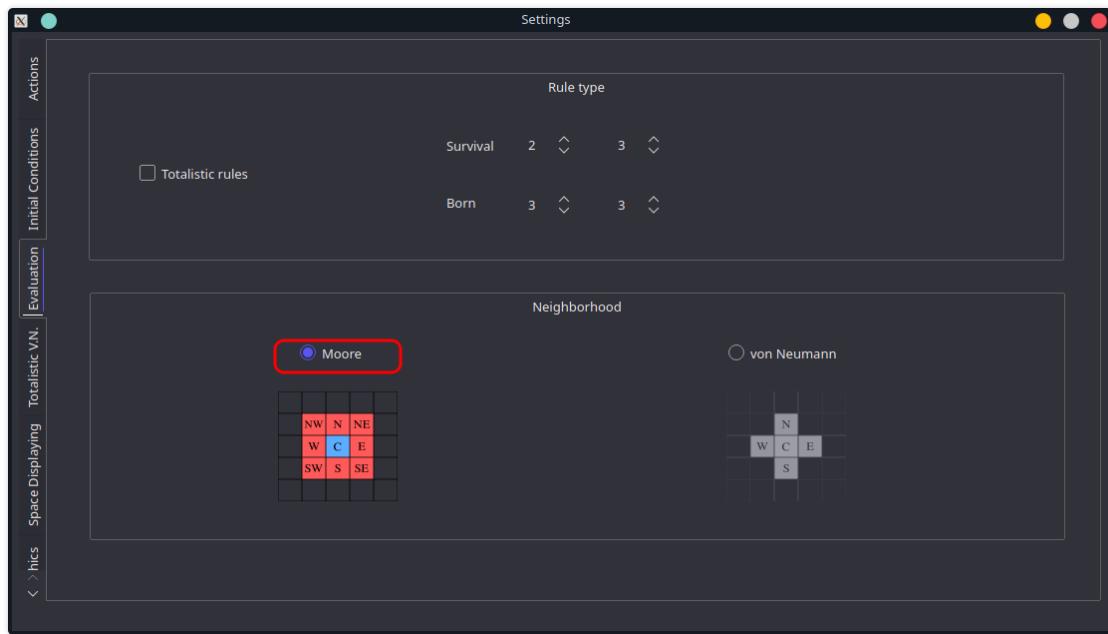


Fig. 6.1.3: IU-CU02 Definir el tipo de vecindad como Moore

6.1.4. IU-CU02.1: Definir el tipo de vecindad como von Neumann

Objetivo: El usuario define el tipo de vecindad con la que se realizará la evaluación del AC como tipo von Neumann

Diseño: Esta pantalla aparece al hacer click en la pestaña **Evaluation** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Ninguna

Salidas: Ninguna

Comandos:

- von Neumann : Define *von Neumann* como el tipo de vecindad con la que se evalúa el AC

Mensajes: Ninguno

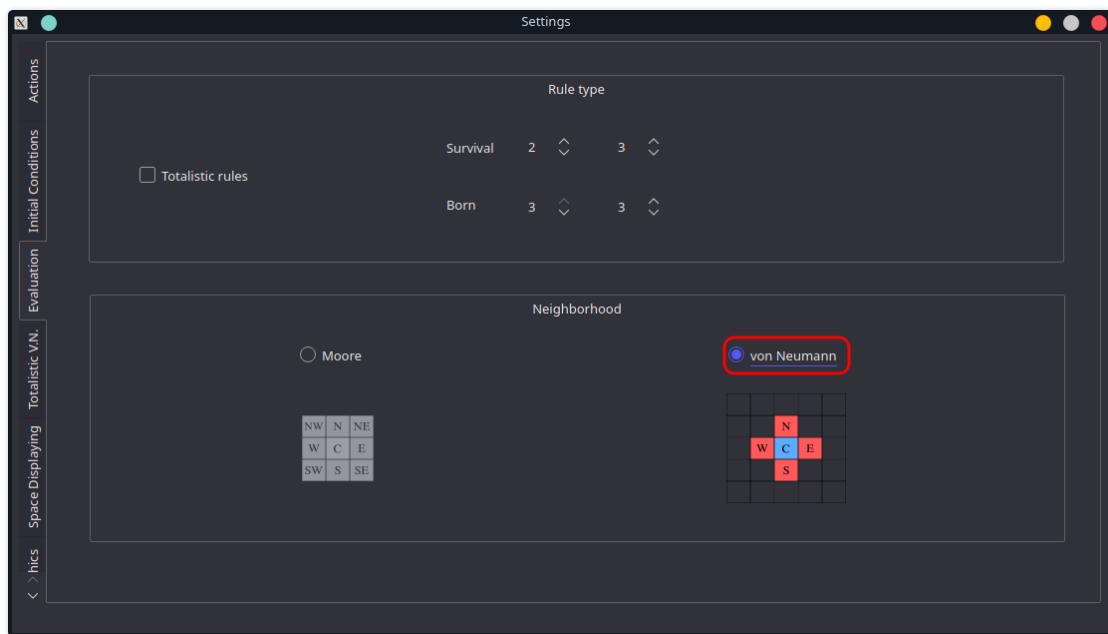


Fig. 6.1.4: IU-CU02.1 Definir el tipo de vecindad como von Neumann

6.1.5. IU-CU03: Definir el tipo de regla como totalista

Objetivo: El usuario define el tipo de regla como totalista, si el tipo de vecindad actual es Moore esta es cambiada al tipo von Neumann.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Evaluation** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Ninguna

Salidas: Ninguna

Comandos: Tipo de regla **Mensajes:** Ninguno

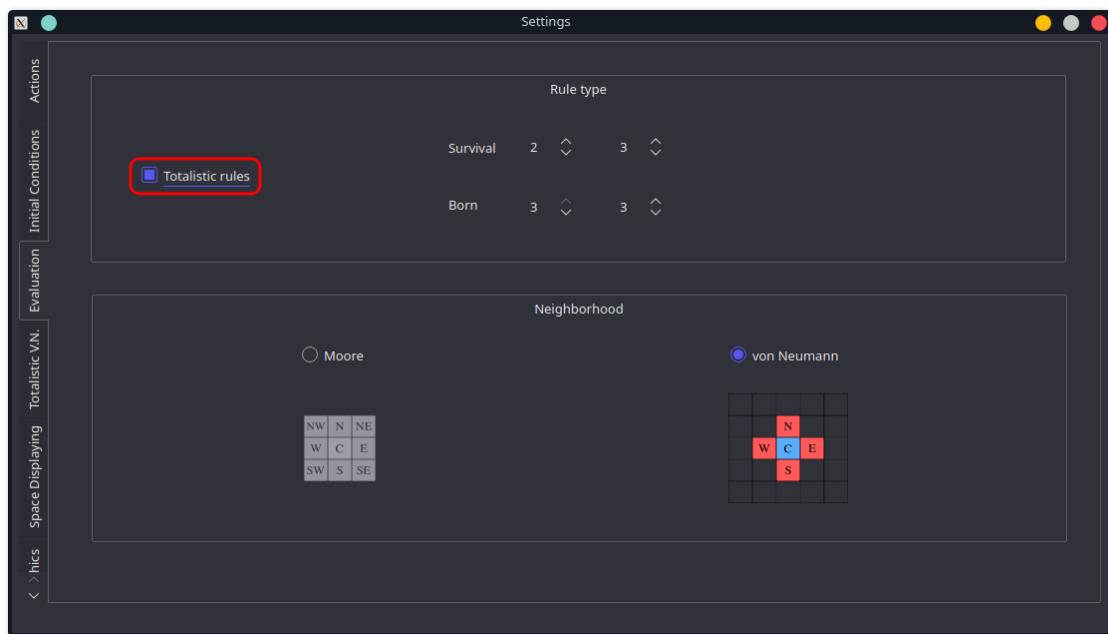


Fig. 6.1.5: IU-CU03 Definir el tipo de regla como totalista

6.1.6. IU-CU04: Cambiar el valor de la célula para una determinada configuración

Objetivo: Las reglas del tipo totalista se definen haciendo click sobre alguno de los 32 botones existentes para establecer si dada una determinada configuración de células, en la siguiente generación la célula actual vive o muere.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Totalistic V.N.** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Máscara de bits de la configuración a cambiar

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

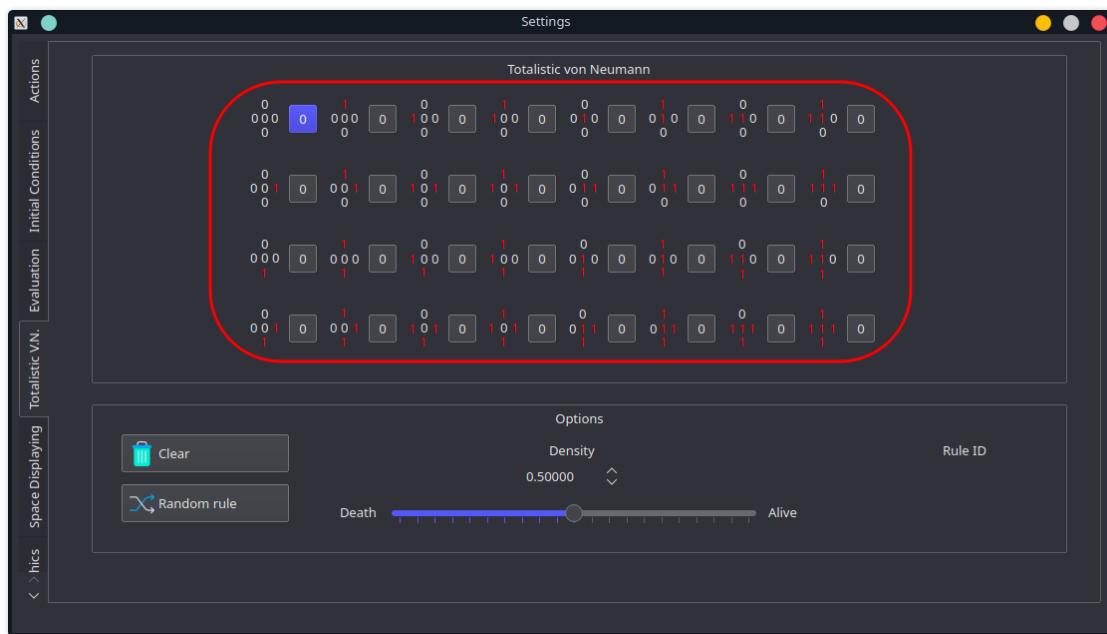


Fig. 6.1.6: IU-CU04 Cambiar el valor de la célula para una determinada configuración

6.1.7. IU-CU04.1: Cambiar la densidad para una regla totalista aleatoria

Objetivo: El usuario puede generar de forma aleatoria una regla de evolución del tipo totalista definiendo la probabilidad de aparición del estado 1.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Totalistic V.N.** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Probabilidad de aparición del estado 1

Salidas: Ninguna

Comandos:

- \wedge : Aumenta la probabilidad de aparición del estado 1
- \vee : Disminuye la probabilidad de aparición del estado 1

Mensajes: Ninguno

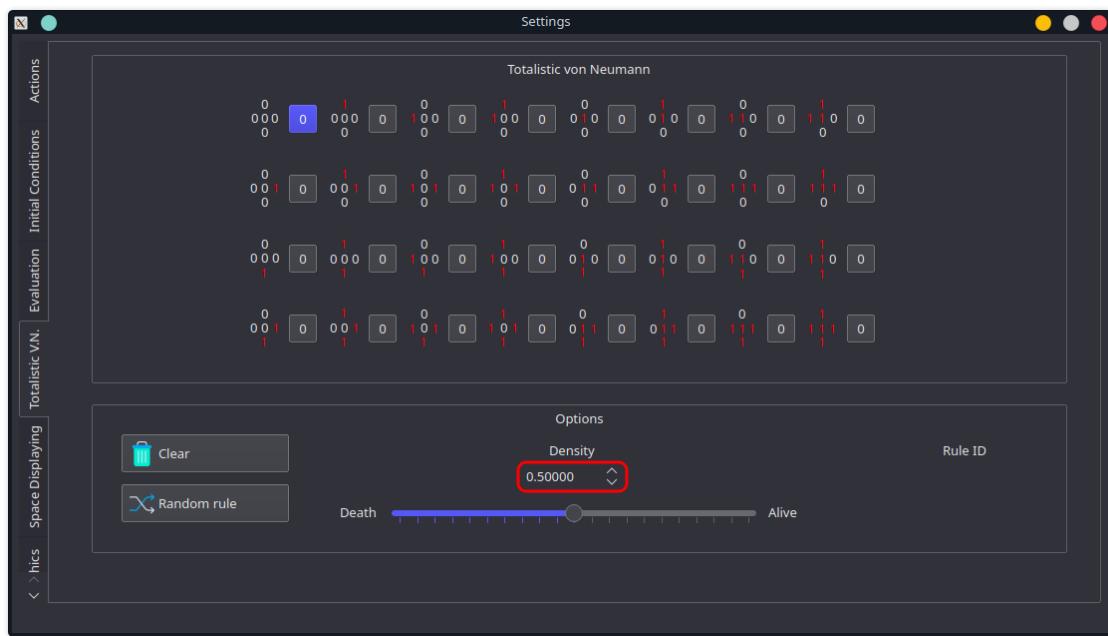


Fig. 6.1.7: IU-CU04.1 Cambiar la densidad para una regla totalista aleatoria

6.1.8. IU-CU04.1-1: Establecer regla aleatoria

Objetivo: Cuando el usuario presiona **Random rule** el sistema obtenga la densidad de aparición del estado 1 y genere aleatoriamente una regla del tipo totalista.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Totalistic V.N.** que se encuentra de lado izquierdo de la ventana principal (*Settings*).

Entradas: Probabilidad de aparición del estado 1.

Salidas: Ninguna

Comandos: Ninguno **Mensajes:** Ninguno

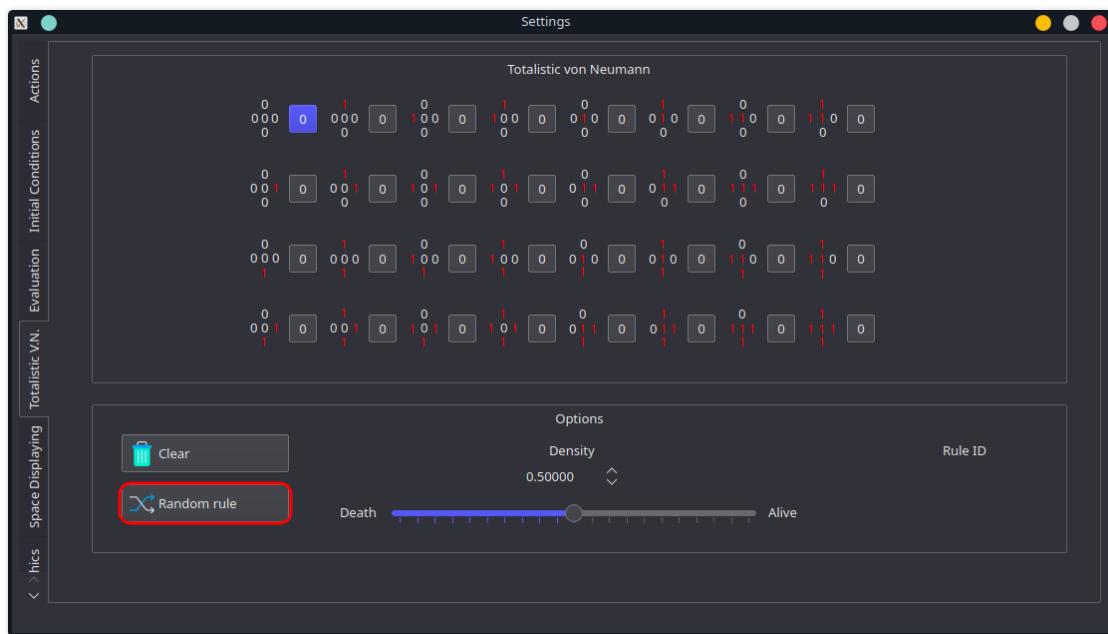


Fig. 6.1.8: IU-CU04.1-1 Establecer regla aleatoria

6.1.9. IU-CU04.2: Ingresar ID de la regla totalista

Objetivo: El usuario ingresa el ID de una regla mediante **ID regla totalista** con lo cual el sistema obtiene una máscara de bits de dicho ID y genera la regla ingresada por el usuario.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Totalistic V.N.** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: ID de una regla

Salidas: Ninguna

Comandos: Ninguno **Mensajes:** Ninguno

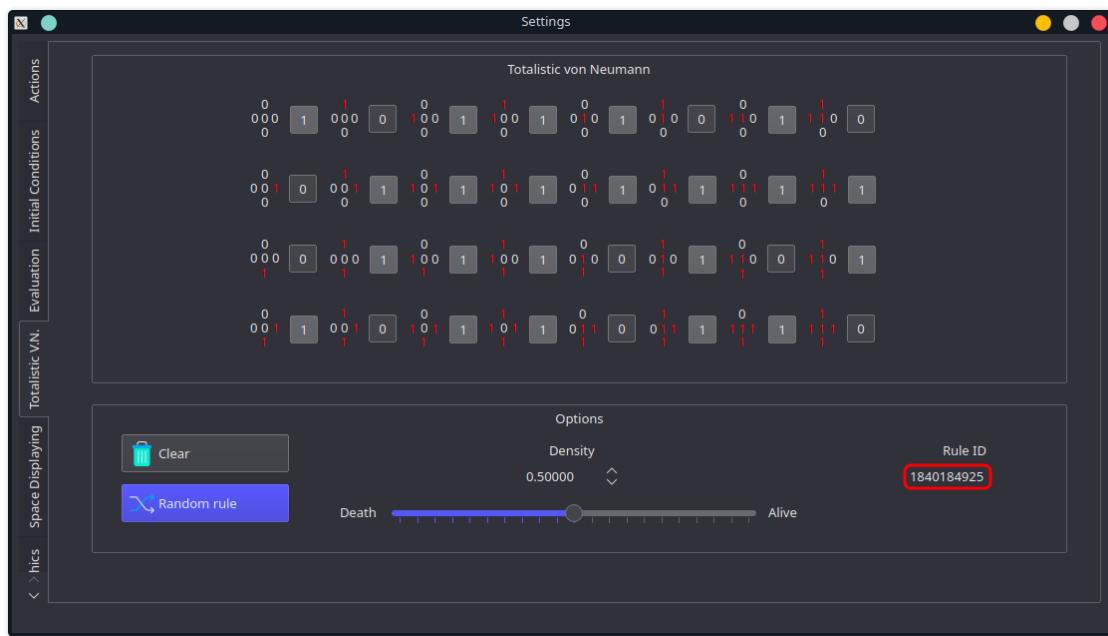


Fig. 6.1.9: IU-CU04.2 Ingresar ID de la regla totalista

6.1.10. IU-CU05 Restablacer valores de la regla totalista

Objetivo: El usuario presiona **Clear** entonces el sistema restablece los valores de los 32 botones para definir la regla a cero de igual forma el ID de la regla cambia a cero.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Totalistic V.N.** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Ninguna

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

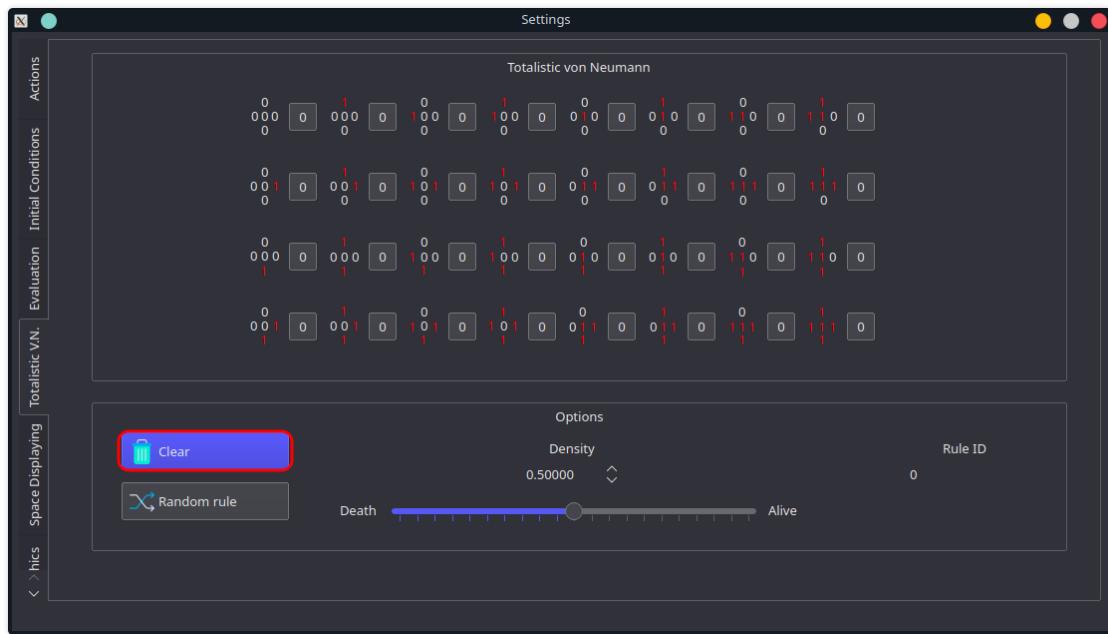


Fig. 6.1.10: IU-CU05 Restablacer valores de la regla totalista

6.1.11. IU-CU06: Definir N_{min}

Objetivo: El usuario definirá el valor de la variable N_{min}

Diseño: Esta pantalla aparece al hacer click en la pestaña **Evaluation** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

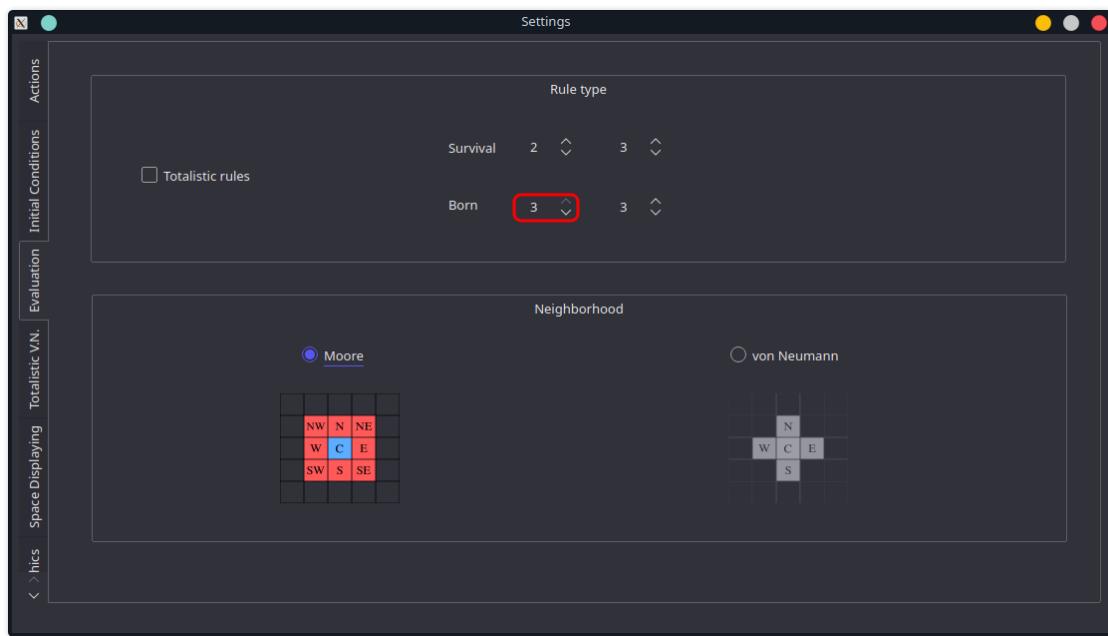
Entradas: N_{min}

Salidas: Ninguna

Comandos:

- \wedge : Aumenta en una unidad el valor de la regla
- \vee : Disminuye en una unidad el valor de la regla

Mensajes: Ninguno

Fig. 6.1.11: IU-CU06 Definir N_{min}

6.1.12. IU-CU06.1: Definir N_{max}

Objetivo: El usuario definirá el valor de la variable N_{max}

Diseño: Esta pantalla aparece al hacer click en la pestaña **Evaluation** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

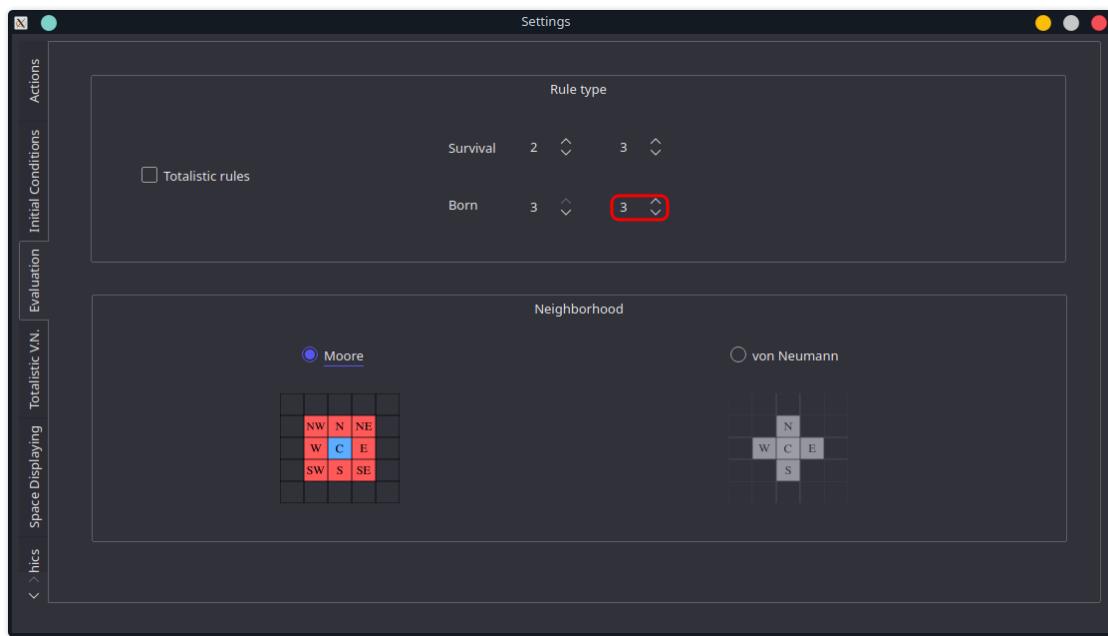
Entradas: N_{max}

Salidas: Ninguna

Comandos:

- \wedge : Aumenta en una unidad el valor de la regla
- \vee : Disminuye en una unidad el valor de la regla

Mensajes: Ninguno

Fig. 6.1.12: IU-CU06.1 Definir N_{max}

6.1.13. IU-CU06.2: Definir S_{min}

Objetivo: El usuario definirá el valor de la variable S_{min}

Diseño: Esta pantalla aparece al hacer click en la pestaña **Evaluation** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

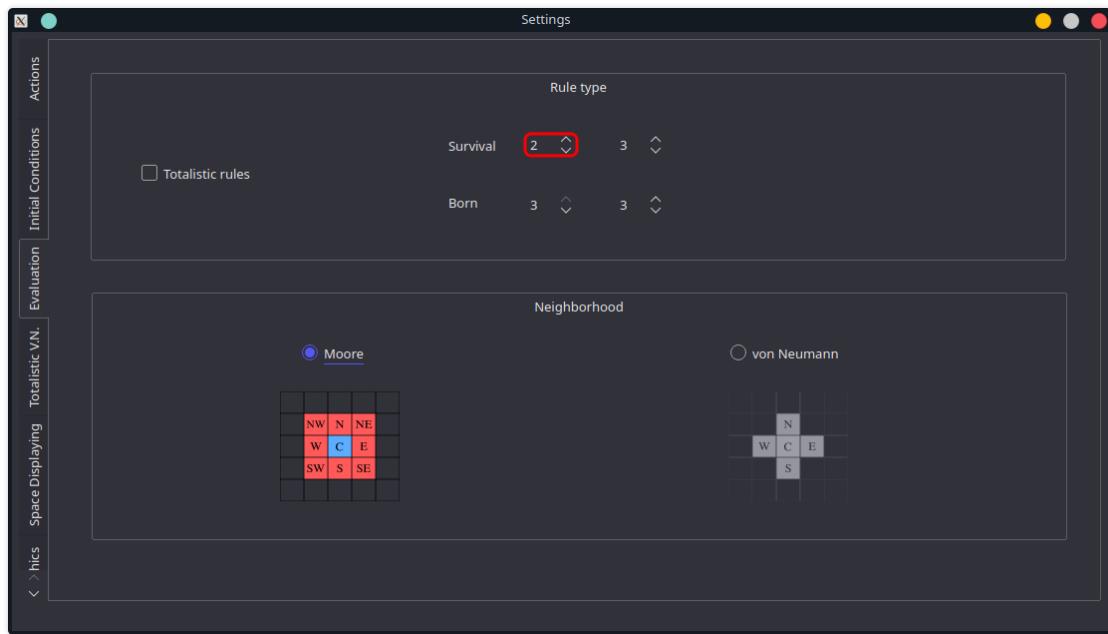
Entradas: S_{min}

Salidas: Ninguna

Comandos:

- \wedge : Aumenta en una unidad el valor de la regla
- \vee : Disminuye en una unidad el valor de la regla

Mensajes: Ninguno

Fig. 6.1.13: IU-CU06.2 Definir S_{min}

6.1.14. IU-CU06.3: Definir S_{max}

Objetivo: El usuario definirá el valor de la variable S_{max}

Diseño: Esta pantalla aparece al hacer click en la pestaña **Evaluation** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

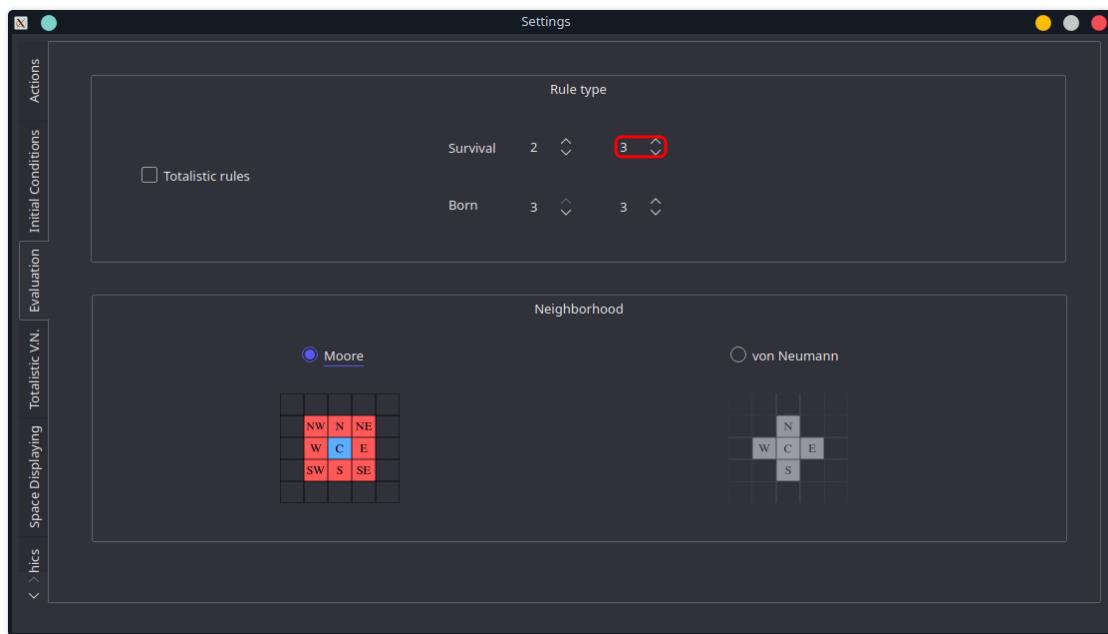
Entradas: S_{max}

Salidas: Ninguna

Comandos:

- \wedge : Aumenta en una unidad el valor de la regla
- \vee : Disminuye en una unidad el valor de la regla

Mensajes: Ninguno

Fig. 6.1.14: IU-CU06.3 Definir S_{max}

6.1.15. IU-CU07: Definir tipo de espacio como cerrado

Objetivo: Al presionar Closed space el espacio de evoluciones será evaluado como si su geométricamente fuese de un plano.

Diseño: Esta pantalla aparece al hacer click en la pestaña Initial Conditions que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Tipo de espacio

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

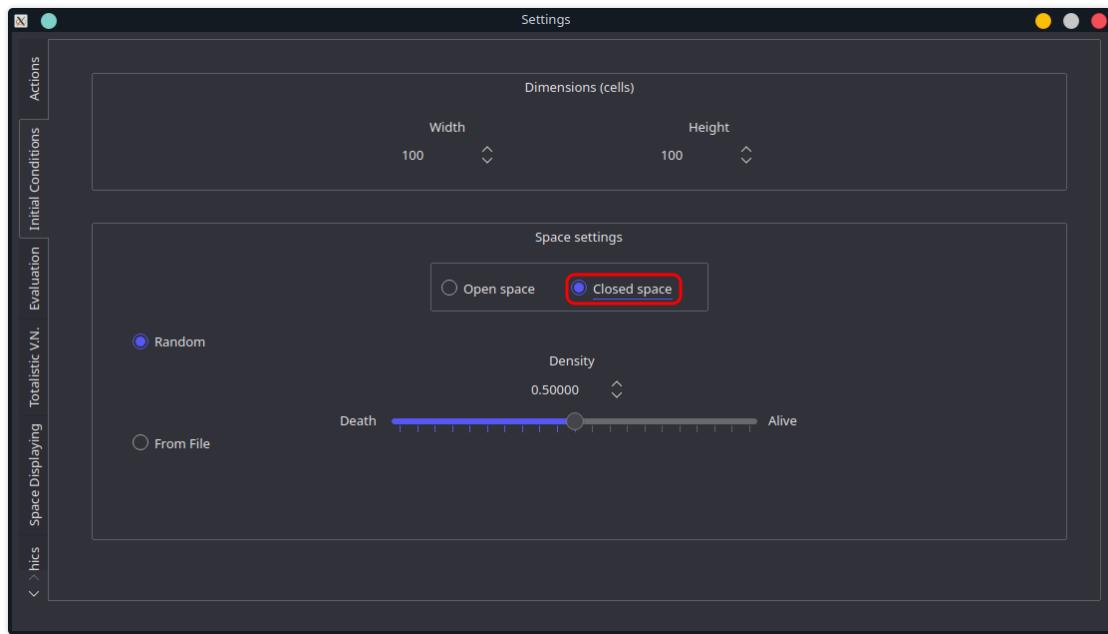


Fig. 6.1.15: IU-CU07 Definir tipo de espacio como cerrado

6.1.16. IU-CU07.1: Definir tipo de espacio como abierto

Objetivo: Al presionar Closed space el espacio de evoluciones será evaluado como si su geométricamente este fuese de un toroide.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Initial Conditions** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Tipo de espacio

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

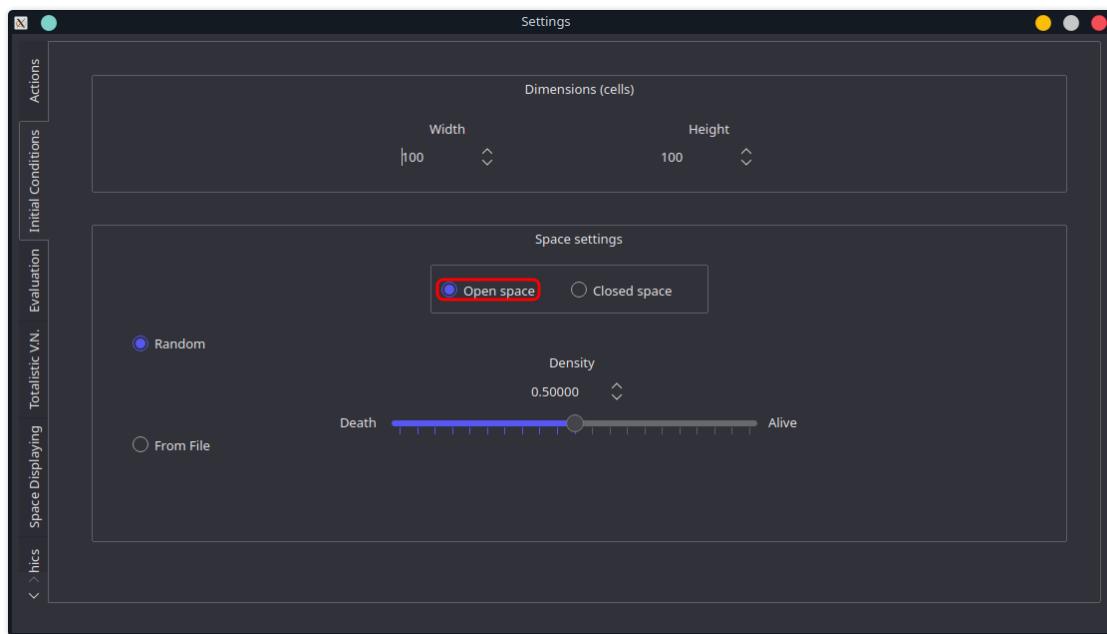


Fig. 6.1.16: IU-CU07.1 Definir tipo de espacio como abierto

6.1.17. IU-CU08: Establecer configuración inicial aleatoria

Objetivo: El usuario definirá la densidad de aparición de las células $c_{i,j}$ cuyo estado es 1

Diseño: Esta pantalla aparece al hacer click en la pestaña **Initial conditions** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Valor de la densidad

Salidas: Ninguna

Comandos:

- Random : Permite que la configuración inicial esté dada de forma aleatoria
- ^ : Aumenta en 0.00001 unidades el valor de la densidad
- v : Disminuye en 0.00001 unidades el valor de la densidad

Mensajes: Ninguno

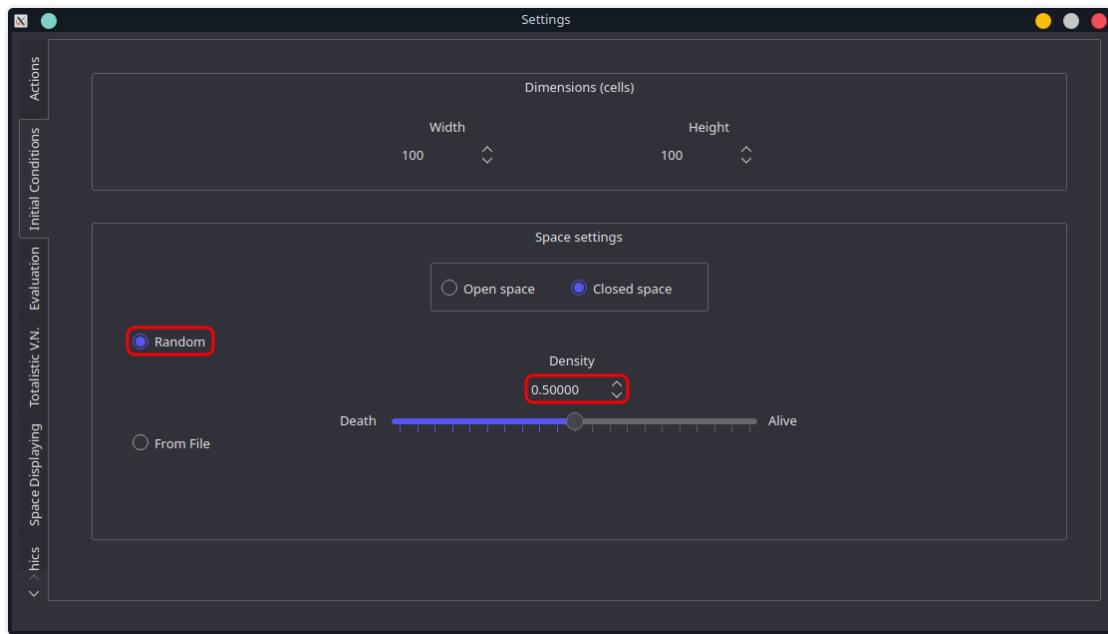


Fig. 6.1.17: IU-CU08 Establecer configuración inicial aleatoria

6.1.18. IU-CU08.1: Establecer configuración inicial desde archivo

Objetivo: El usuario obtiene desde un archivo alguna configuración con la cual puede ser inicializado el AC, esto incluye las células $c_{i,j}$ cuyo estado es 1

Diseño: Esta pantalla aparece al hacer click en la pestaña **Initial conditions** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Archivo con la configuración inicial

Salidas: Ninguna

Comandos: From File : Permite que la configuración inicial sea cargada desde un archivo **Mensajes:** Ninguno

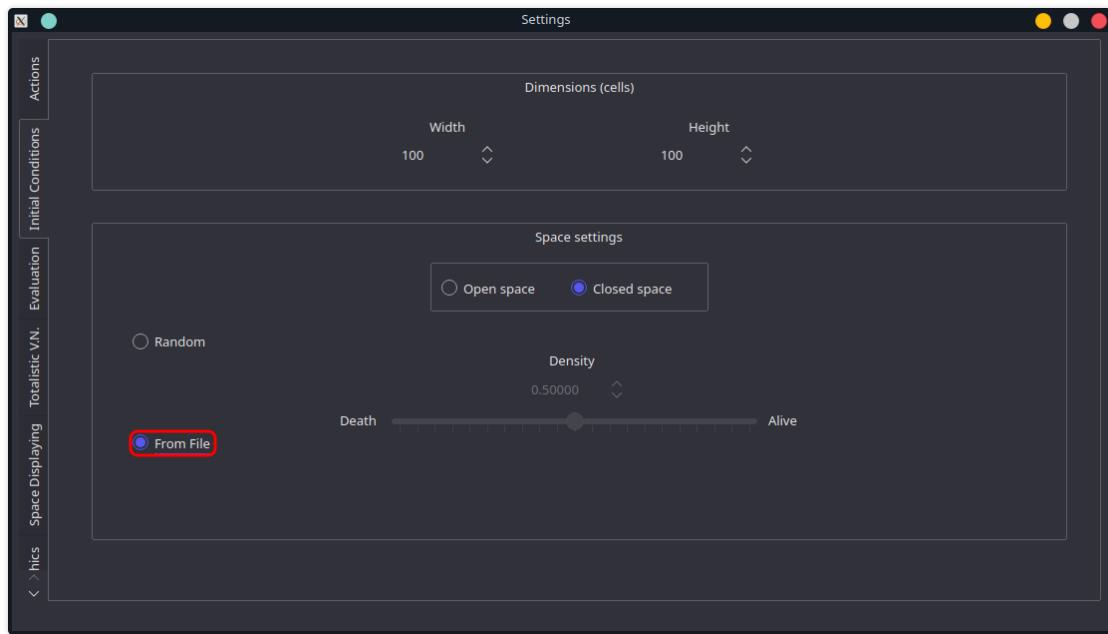


Fig. 6.1.18: IU-CU08.1 Establecer configuración inicial desde archivo

6.1.19. IU-CU08.1-1: Importar archivo

Objetivo: El usuario da click en **Import CA** esto despliega una ventana la cual permite seleccionar un archivo que contiene un objeto CA

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Archivo con la configuración inicial

Salidas: Ninguna

Comandos: Ninguno **Mensajes:** Ninguno

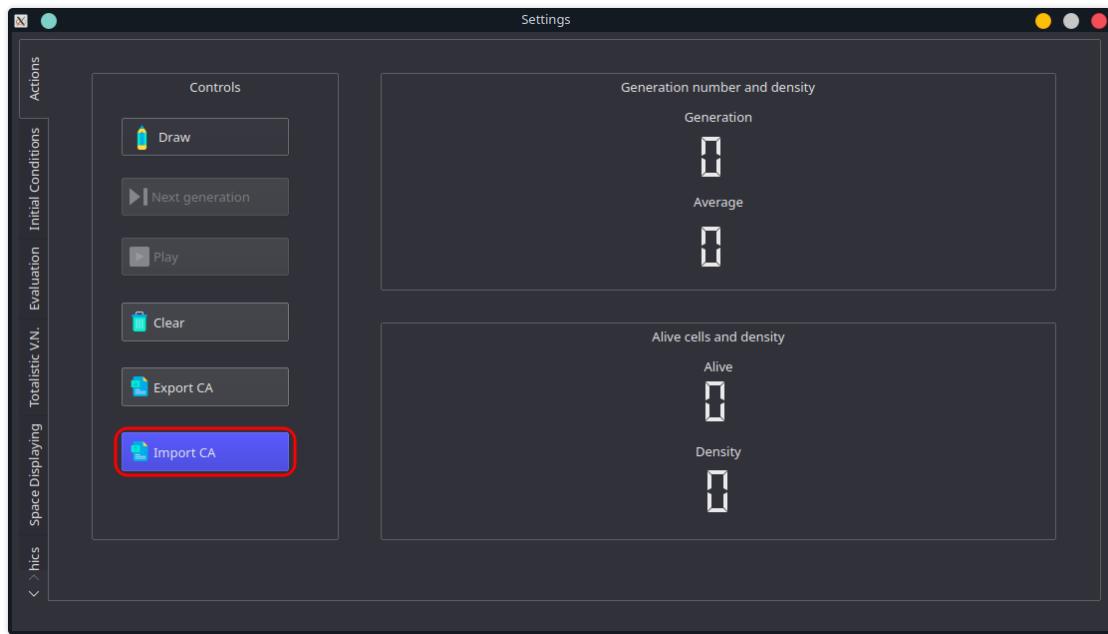


Fig. 6.1.19: IU-CU08.1-1 Importar archivo

6.1.20. IU-CU08.1-2: Ventana de selección de archivo

Objetivo: El usuario selecciona un archivo que contenga la configuración inicial que desee cargar en el simulador

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** vía la IU-CU08.1-1 Importar archivo

Entradas: Ruta del archivo que contiene la configuración inicial

Salidas: Ninguna

Comandos:

- Abrir : Selecciona el archivo y carga el archivo en memoria
- Cancelar : Cierra la ventana de selección de archivo sin cargar el archivo

Mensajes: Ninguno

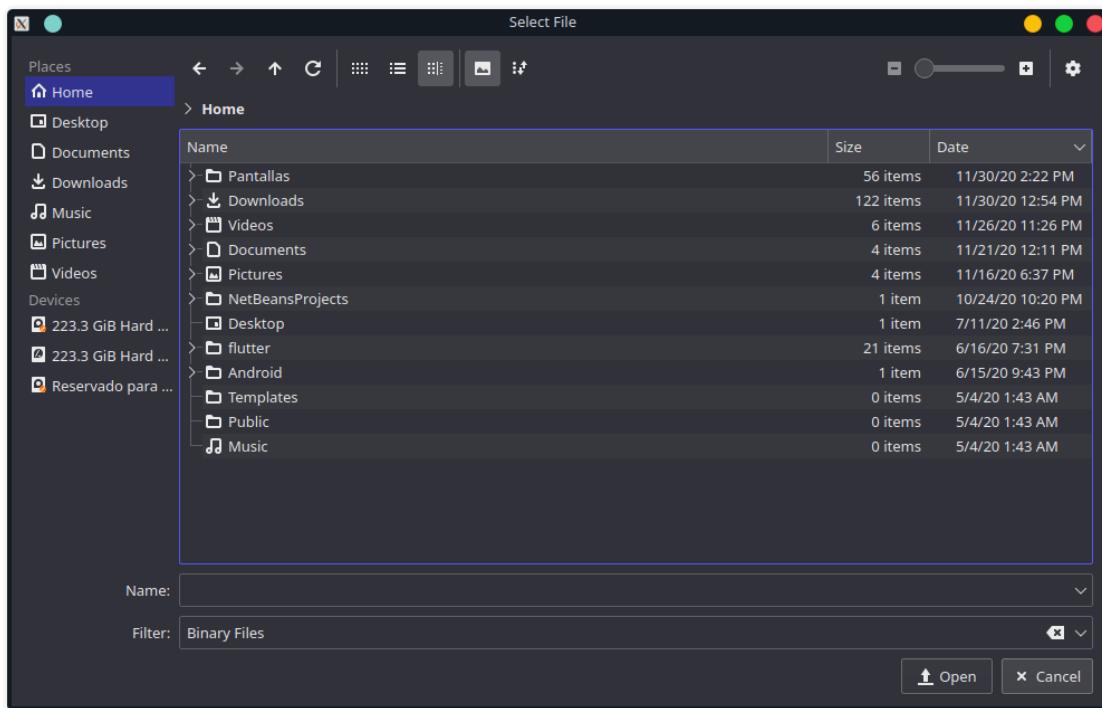


Fig. 6.1.20: IU-CU08.1-2 Ventana de selección de archivo

6.1.21. IU-CU09: Exportar objeto AC

Objetivo: El usuario presiona **Export CA** esto despliega una ventana en la cual puede seleccionar el directorio donde el objeto será almacenado.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions**

Entradas: Ninguna

Salidas: Ninguna

Comandos: Ninguno **Mensajes:** Ninguno

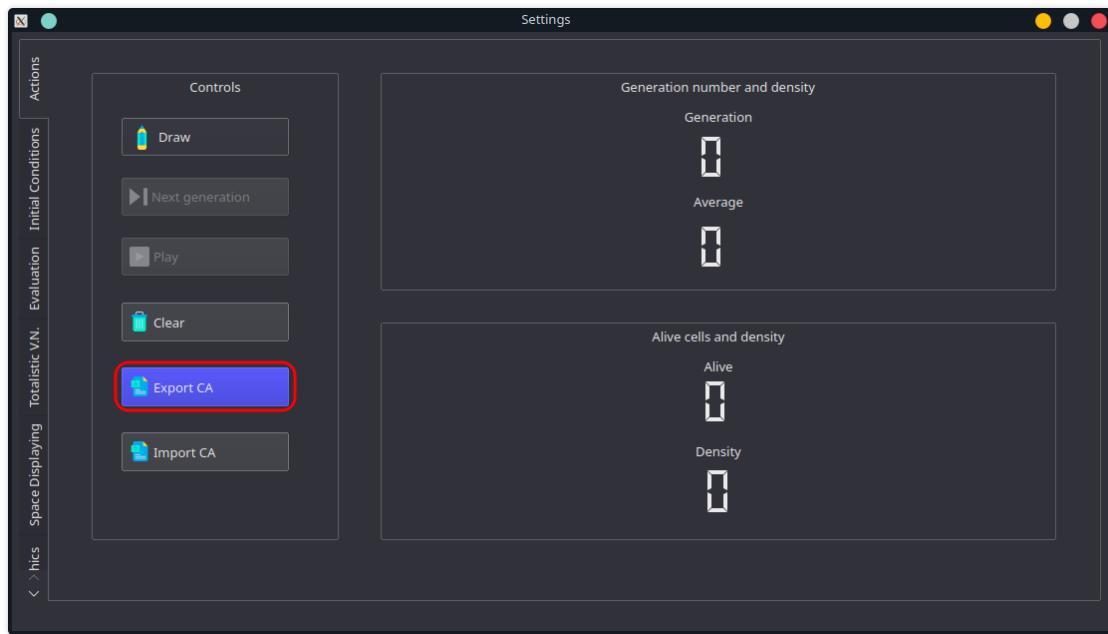


Fig. 6.1.21: IU-CU09 Exportar objeto AC.

6.1.22. IU-CU09.1: Seleccionar directorio

Objetivo: El usuario selecciona un directorio donde el objeto AC será almacenado.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** vía la IU-CU09 Exportar objeto AC

Entradas: Ruta del directorio donde se almacenará el objeto AC

Salidas: Ninguna

Comandos:

- Guardar : Almacena el objeto AC.
- Cancelar : Cierra la ventana de selección de archivo sin almacenar el objeto AC.

Mensajes: Ninguno

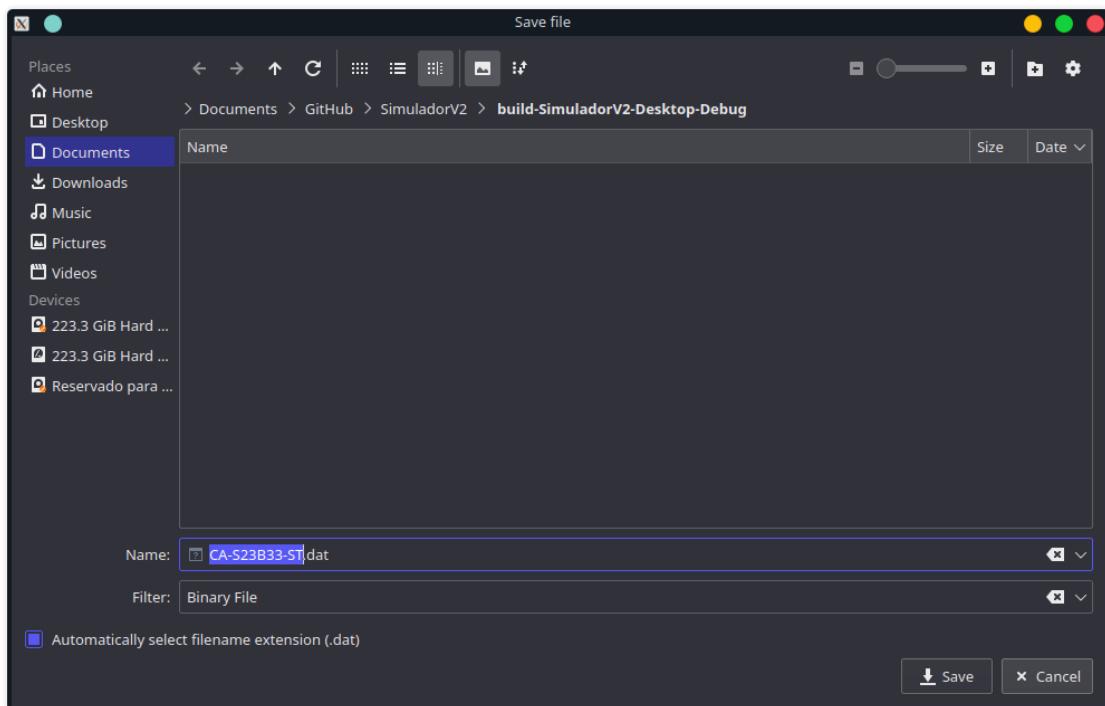


Fig. 6.1.22: IU-CU09.1 Seleccionar directorio

6.1.23. IU-CU10: Definir el tamaño de las células

Objetivo: El usuario definirá el tamaño de las células en pixeles

Diseño: Esta pantalla aparece al hacer click en la pestaña **Space Displaying** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Tamaño de las células

Salidas: Ninguna

Comandos:

- ^ : Aumenta en una unidad el tamaño de las células
- v : Disminuye en una unidad el tamaño de las células

Mensajes: Ninguno

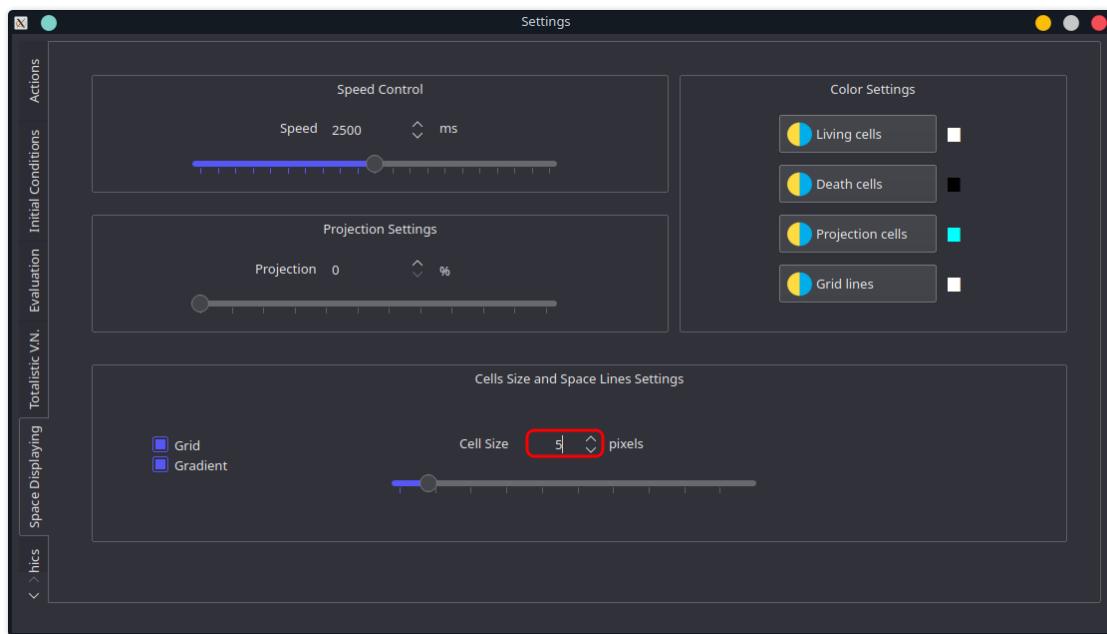


Fig. 6.1.23: IU-CU10 Definir el tamaño de las células

6.1.24. IU-CU11: Definir la velocidad de evolución del AC

Objetivo: El usuario define la velocidad con la que evolucionará el AC en la $t, t+1, \dots, t+n$ generación

Diseño: Esta pantalla aparece al hacer click en la pestaña **Space Displaying** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Valor de la velocidad

Salidas: Ninguna

Comandos:

- \wedge : Aumenta en una unidad el valor de la velocidad
- \vee : Disminuye en una unidad el valor de la velocidad

Mensajes: Ninguno

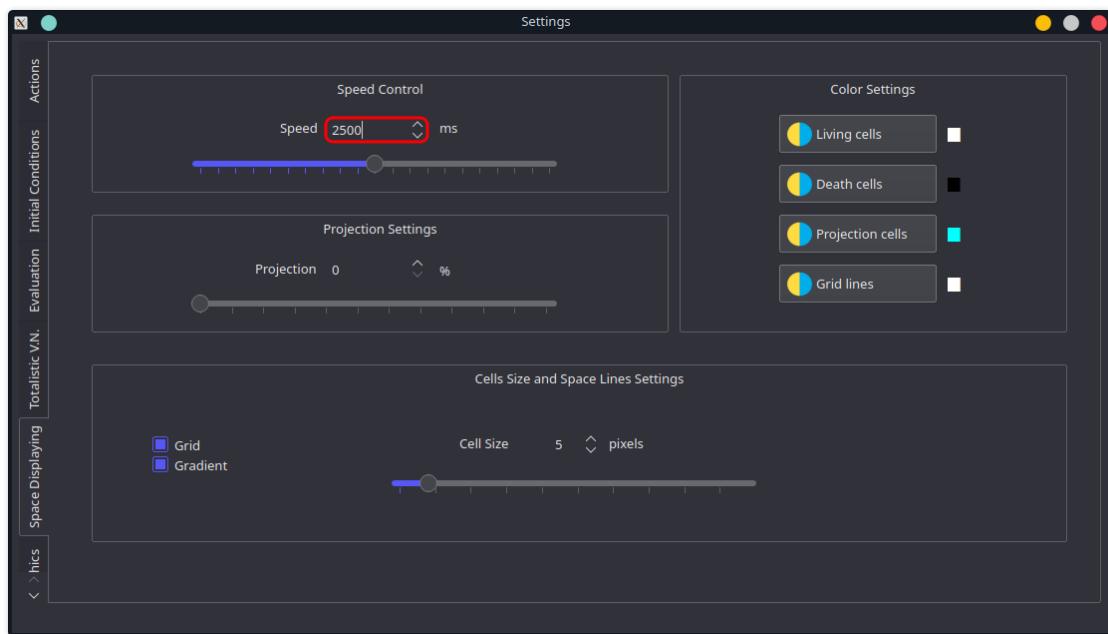


Fig. 6.1.24: IU-CU11 Definir la velocidad de evolución del AC

6.1.25. IU-CU12: Asignar valor del escalar

Objetivo: El usuario ingresa el valor de un escalar el cual permite redimensionar la submatriz del espacio de evoluciones que es estática y será evaluada normalmente mientras que el resto del espacio de evoluciones se proyecta en el eje z

Diseño: Esta pantalla aparece al hacer click en la pestaña **Space displaying** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Valor del escalar

Salidas: Ninguna

Comandos:

- \wedge : Aumenta en una unidad el valor del escalar
- \vee : Disminuye en una unidad el valor del escalar

Mensajes: Ninguno

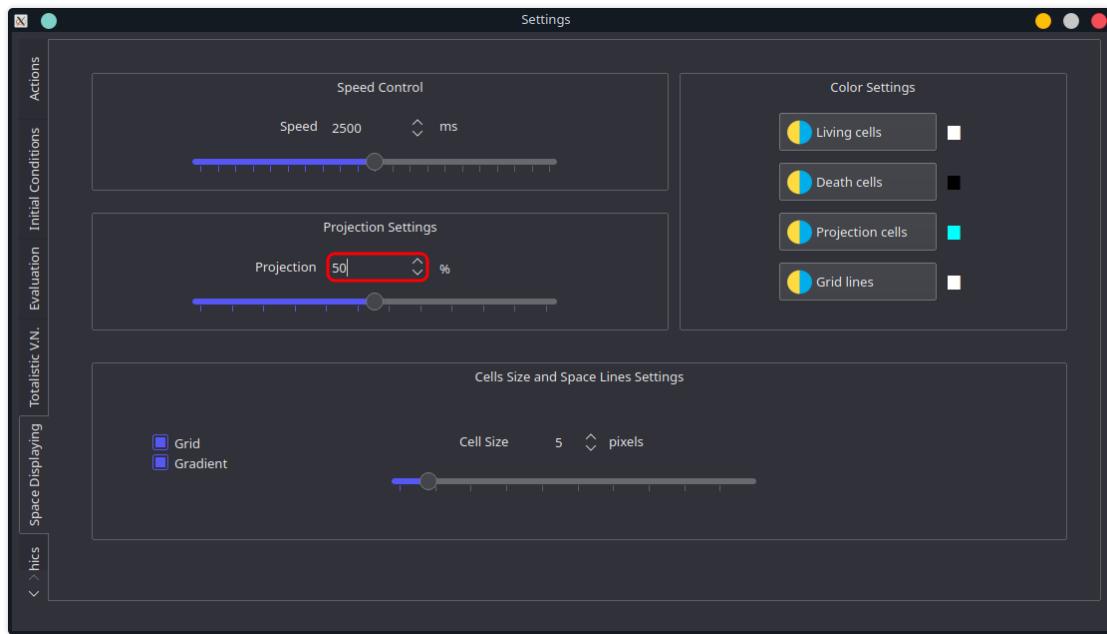


Fig. 6.1.25: IU-CU12 Asignar valor del escalar

6.1.26. IU-CU13: Estado del gradiente

Objetivo: El usuario habilita o deshabilita el gradiente del espacio de evoluciones.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Space displaying** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Estado del gradiente

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Gradient: habilita o deshabilita el gradiente del espacio de evoluciones.

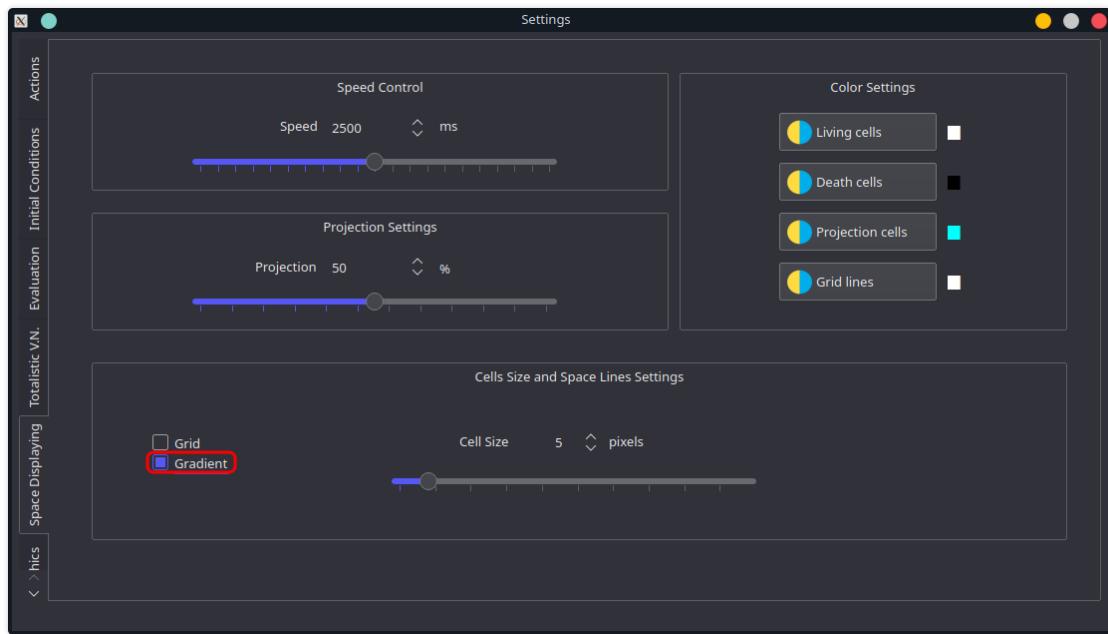


Fig. 6.1.26: IU-CU13 Estado del gradiente.

6.1.27. IU-CU14: Estado de la rejilla

Objetivo: El usuario muestra u oculta la rejilla del espacio de evoluciones.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Space displaying** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Estado de la rejilla

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Grid: habilita o deshabilita la rejilla del espacio de evoluciones.

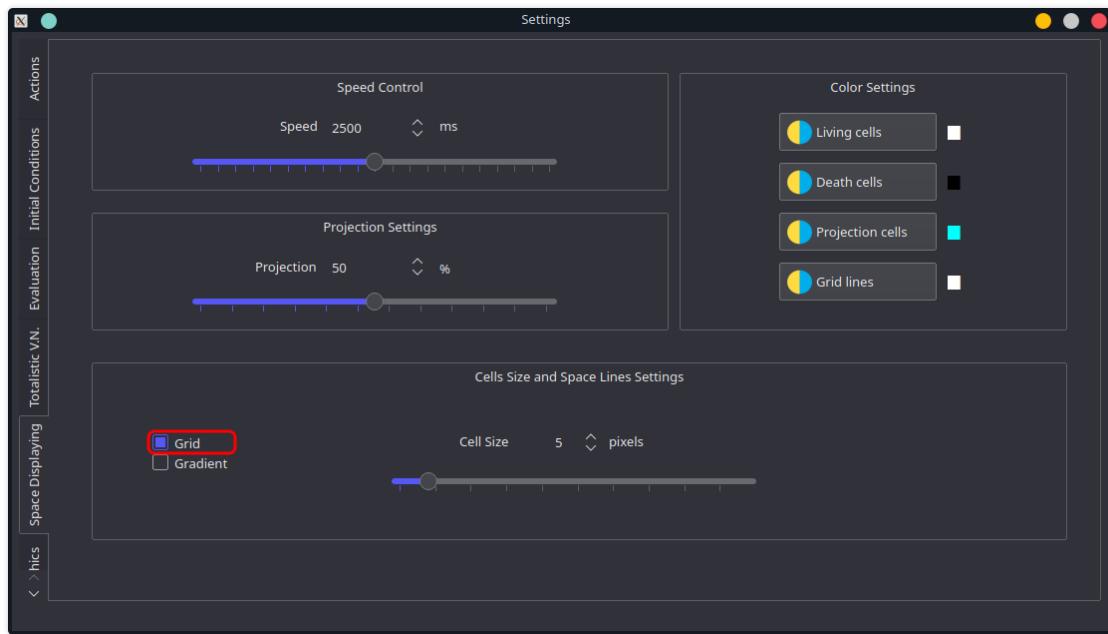


Fig. 6.1.27: IU-CU14 Estado de la rejilla

6.1.28. IU-CU14-1: Espacio de evolución del AC sin rejilla

Objetivo: Mostrar el espacio de evoluciones sin rejilla.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se encuentra de lado izquierdo de la ventana principal (*Settings*) y posteriormente al hacer click en **Draw**

Entradas: Ninguna

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

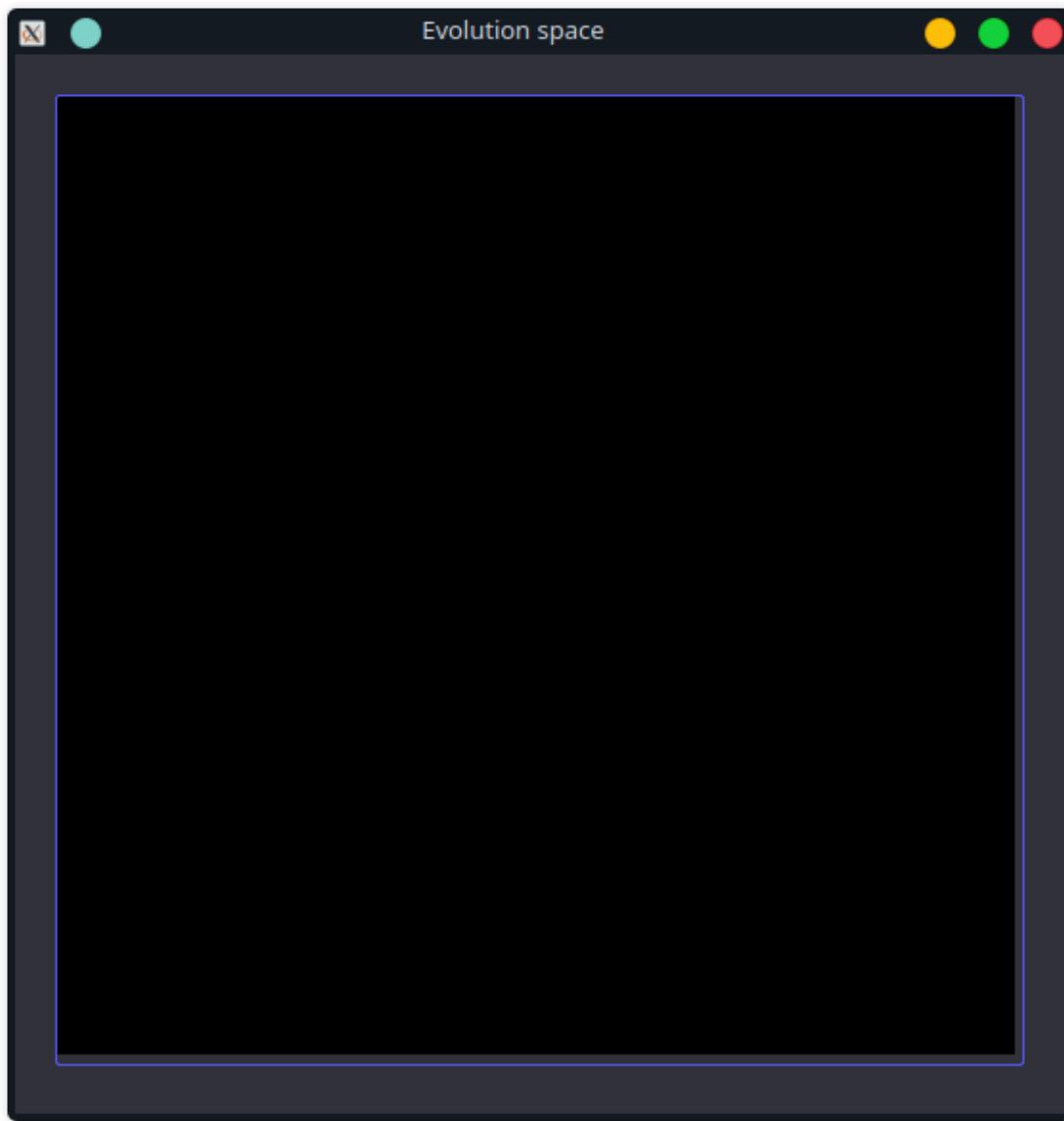


Fig. 6.1.28: IU-CU14-1 Espacio de evolución del AC

6.1.29. IU-CU15: Definir el color de la rejilla

Objetivo: El usuario define el color de la rejilla presionando **Grid lines** lo que despliega una ventana donde elegirá el color deseado.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Space Displaying** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Color de la rejilla

Salidas: Ninguna

Comandos:

- Grid lines : Abre la paleta de colores IU-CU15-1 Paleta de colores

Mensajes: Ninguno

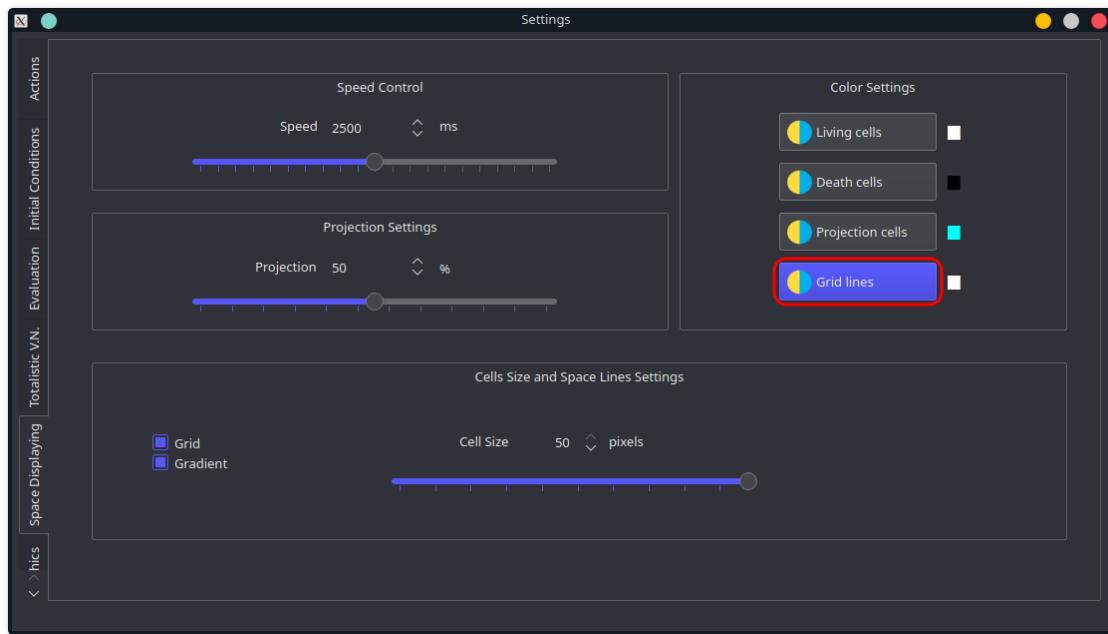


Fig. 6.1.29: IU-CU15 Definir el color de la rejilla

6.1.30. IU-CU15.1: Etiqueta indicadora de la rejilla

Objetivo: Con el fin de facilitar la visualización de los colores al usuario, hay una etiqueta que muestra el color actual de la rejilla.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Space Displaying** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Ninguna

Salidas: Ninguna

Comandos: Ninguno **Mensajes:** Ninguno

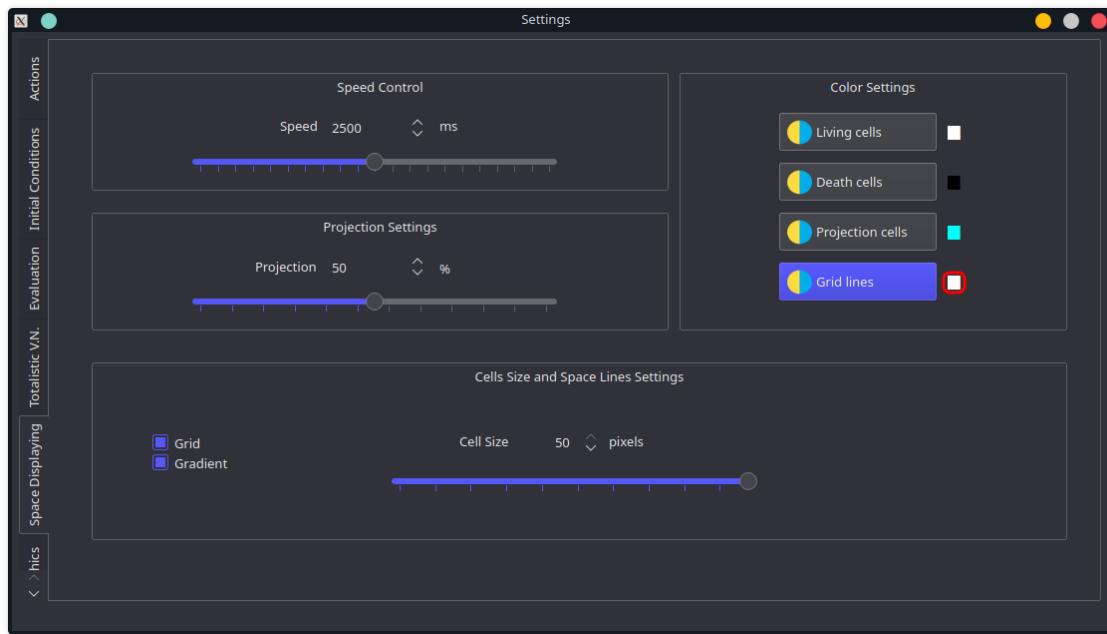


Fig. 6.1.30: IU-CU15.1 Etiqueta indicadora de la rejilla

6.1.31. IU-CU15-1: Paleta de colores

Objetivo: El usuario ingresa o selecciona el color desde la paleta

Diseño: Esta pantalla aparece al hacer click en **Living cells**, **Death cells**, **Projection cells** ó en **Grid lines** vía IU-CU16 Definir color de las células vivas, IU-CU16.1 Definir color de las células muertas, IU-CU16.2 Definir color de las células de proyección, IU-CU15 Definir el color de la rejilla

Entradas: Color de las células o de la rejilla

Salidas: Ninguna

Comandos:

- **Pick Screen Color** : Permite seleccionar cualquier color que se muestre sobre la pantalla
- **Add Custom Colors** : Permite agregar a la paleta un color personalizado
- **Aceptar** : Define el color de la célula
- **Cancelar** : Cierra la paleta de colores sin cambiar el color de la célula
- **Hue**: Permite modificar el tono
- **Sat**: Permite modificar la saturación
- **Val**: Permite modificar la iluminación
- **Red**: Permite modificar el valor del color rojo (0-255)
- **Green**: Permite modificar el valor del color verde (0-255)
- **Blue**: Permite modificar el valor del color azul (0-255)
- **HTML**: Permite introducir un color en formato hexadecimal

Mensajes: Ninguno

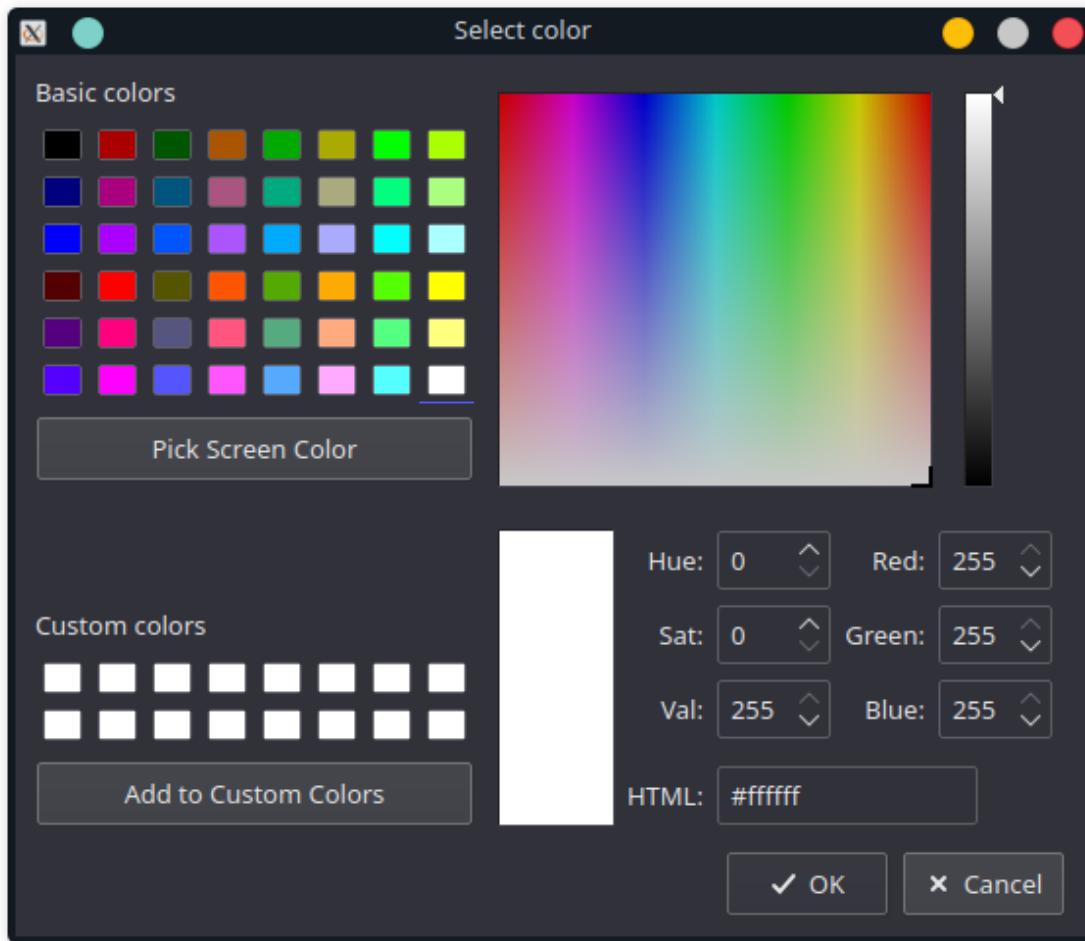


Fig. 6.1.31: IU-CU15-1 Paleta de colores

6.1.32. IU-CU15.2: Rejilla en el espacio de evolución del AC

Objetivo: La rejilla es mostrada u ocultada del espacio de evolución del AC.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se encuentra de lado izquierdo de la ventana principal (*Settings*) y posteriormente al hacer click en **Draw**

Entradas: Estado de la rejilla

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno



Fig. 6.1.32: IU-CU15.2 Rejilla en el espacio de evolución del AC

6.1.33. IU-CU16: Definir el color de las células vivas

Objetivo: El usuario definirá el color de las células vivas

Diseño: Esta pantalla aparece al hacer click en la pestaña **Space Displaying** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Color de las células vivas

Salidas: Ninguna

Comandos:

- Living cells : Abre la paleta de colores IU-CU15-1 Paleta de colores

Mensajes: Ninguno

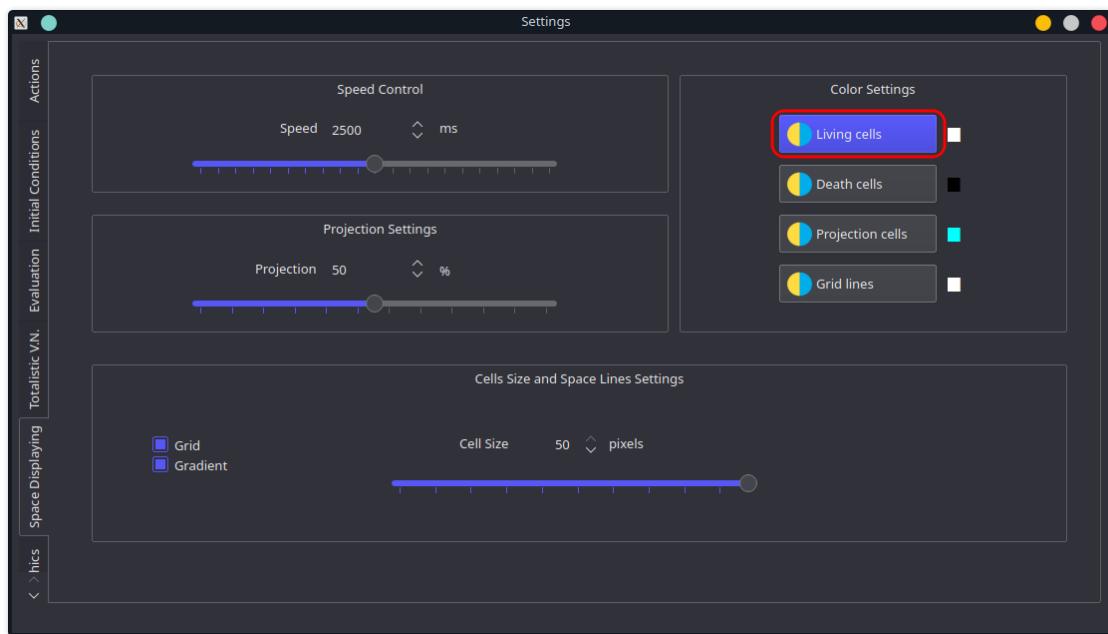


Fig. 6.1.33: IU-CU16 Definir el color de las células vivas

6.1.34. IU-CU16.1: Definir el color de las células muertas

Objetivo: El usuario definirá el color de las células muertas

Diseño: Esta pantalla aparece al hacer click en la pestaña **Space Displaying** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Color de las células muertas

Salidas: Ninguna

Comandos:

- Death cells : Abre la paleta de colores IU-CU15-1 Paleta de colores

Mensajes: Ninguno

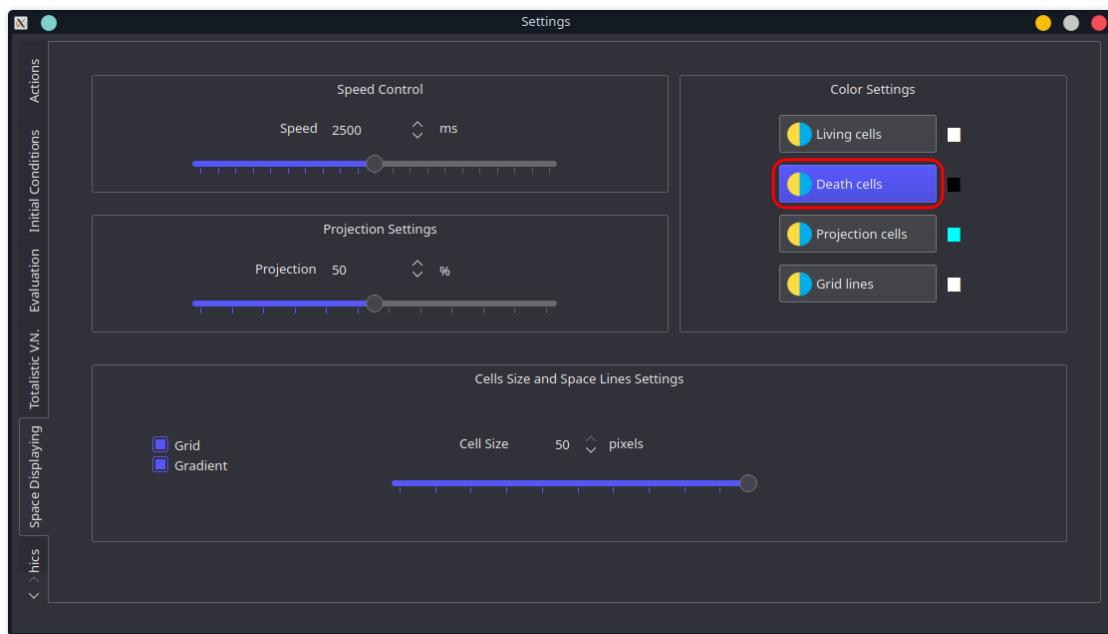


Fig. 6.1.34: IU-CU16.1 Definir el color de las células muertas

6.1.35. IU-CU16.2: Definir el color de las células de proyección

Objetivo: El usuario definirá el color de las células de proyección

Diseño: Esta pantalla aparece al hacer click en la pestaña **Space Displaying** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Color de las células de proyección

Salidas: Ninguna

Comandos:

- **Projection cells :** Abre la paleta de colores IU-CU15-1 Paleta de colores

Mensajes: Ninguno

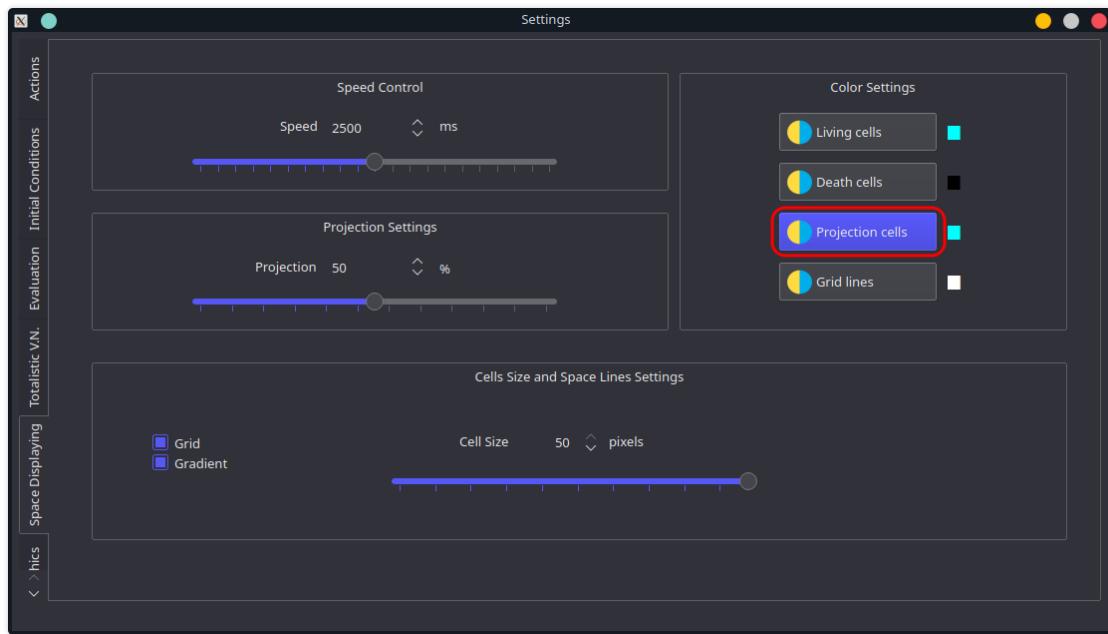


Fig. 6.1.35: IU-CU16.2 Definir el color de las células de proyección

6.1.36. IU-CU16.3: Etiqueta indicadora de las células

Objetivo: Con el fin de facilitar la visualización de los colores al usuario, hay tres etiquetas que muestra el color actual de la células vivas, células muertas y células de proyección.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Space Displaying** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Ninguna

Salidas: Ninguna

Comandos: Ninguno **Mensajes:** Ninguno

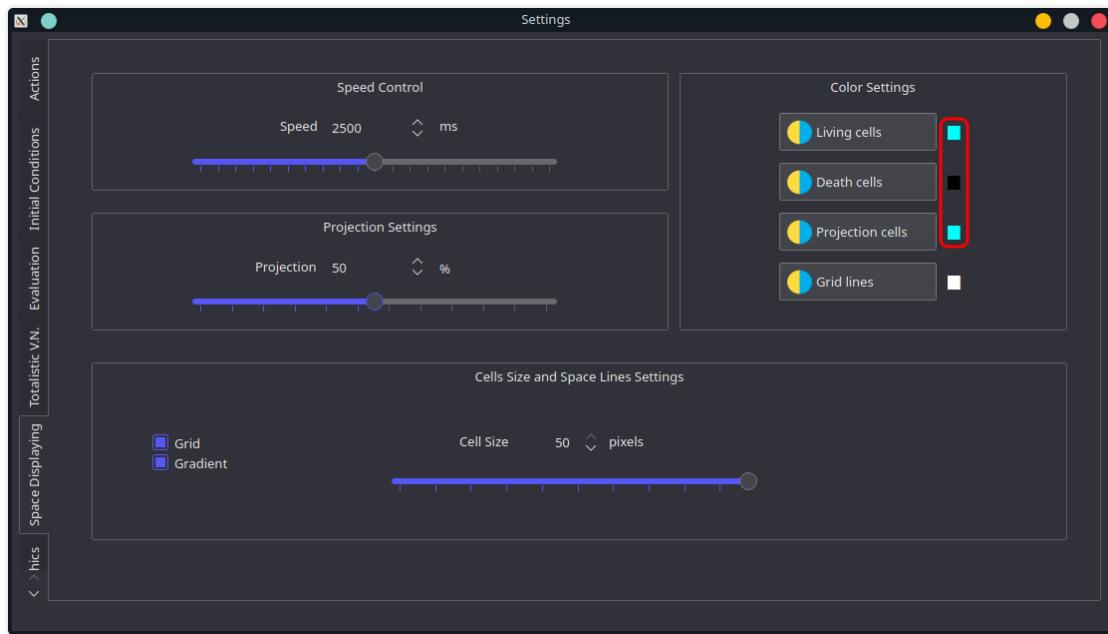


Fig. 6.1.36: IU-CU16.3 Etiqueta indicadora de colores de las células

6.1.37. IU-CU16-1: Células vivas en el espacio de evolución del AC

Objetivo: De acuerdo al color seleccionado por el usuario las células vivas son mostradas en el espacio de evolución

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se encuentra de lado izquierdo de la ventana principal (*Settings*) y posteriormente al hacer click en **Draw**

Entradas: Color de las células vivas

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

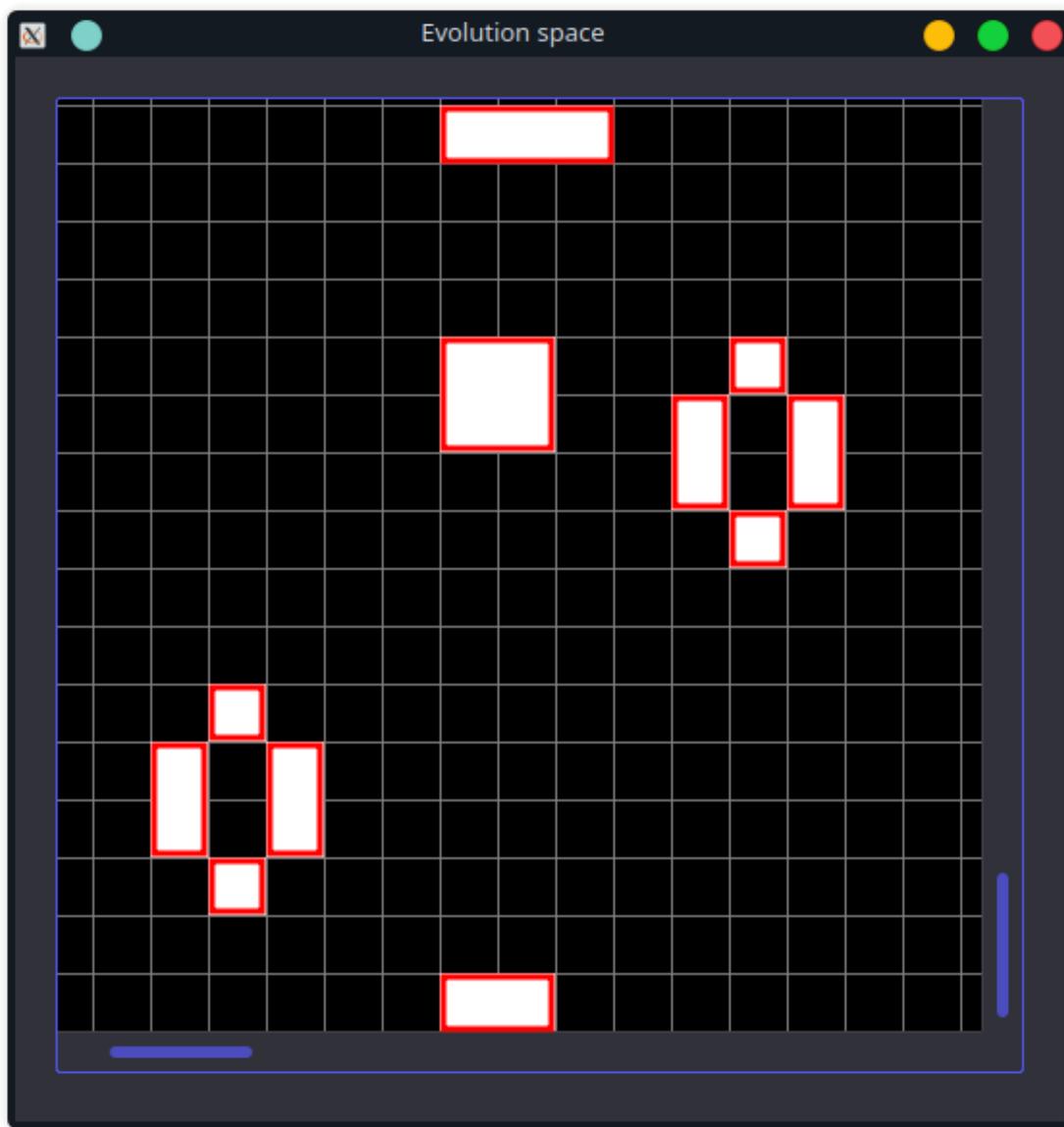


Fig. 6.1.37: IU-CU16-1 Células vivas en el espacio de evolución del AC

6.1.38. IU-CU16-2: Células muertas en el espacio de evolución del AC

Objetivo: De acuerdo al color seleccionado por el usuario las células muertas son mostradas en el espacio de evolución

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se encuentra de lado izquierdo de la ventana principal (*Settings*) y posteriormente al hacer click en **Draw**

Entradas: Color de las células muertas

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

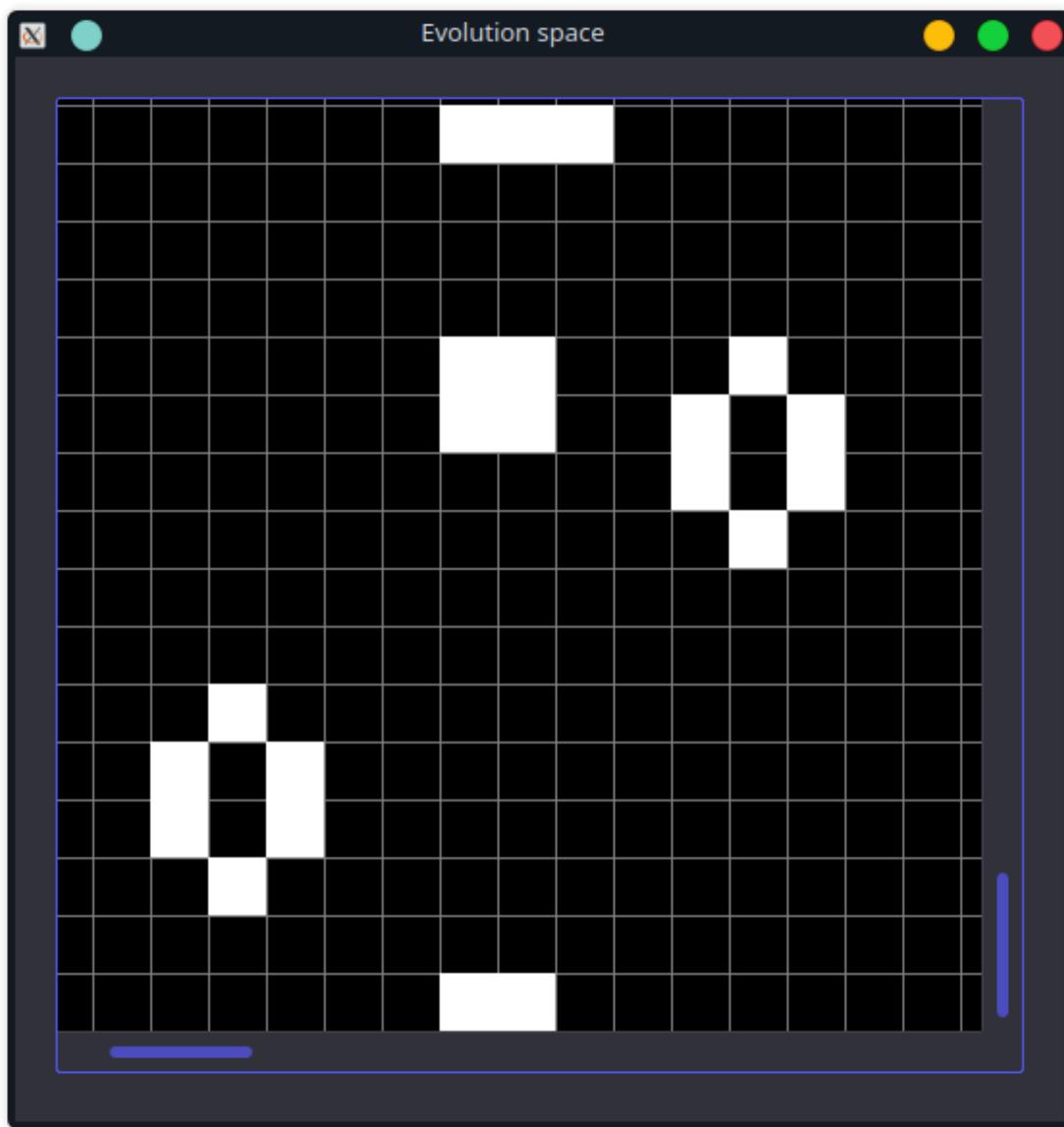


Fig. 6.1.38: IU-CU16-2 Células muertas en el espacio de evolución del AC

6.1.39. IU-CU16-3: Células de proyección en el espacio de evolución del AC

Objetivo: De acuerdo al color seleccionado por el usuario las células de proyección son mostradas en el espacio de evolución

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se encuentra de lado izquierdo de la ventana principal (*Settings*) y posteriormente al hacer click en **Draw**

Entradas: Color de las células de proyección

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

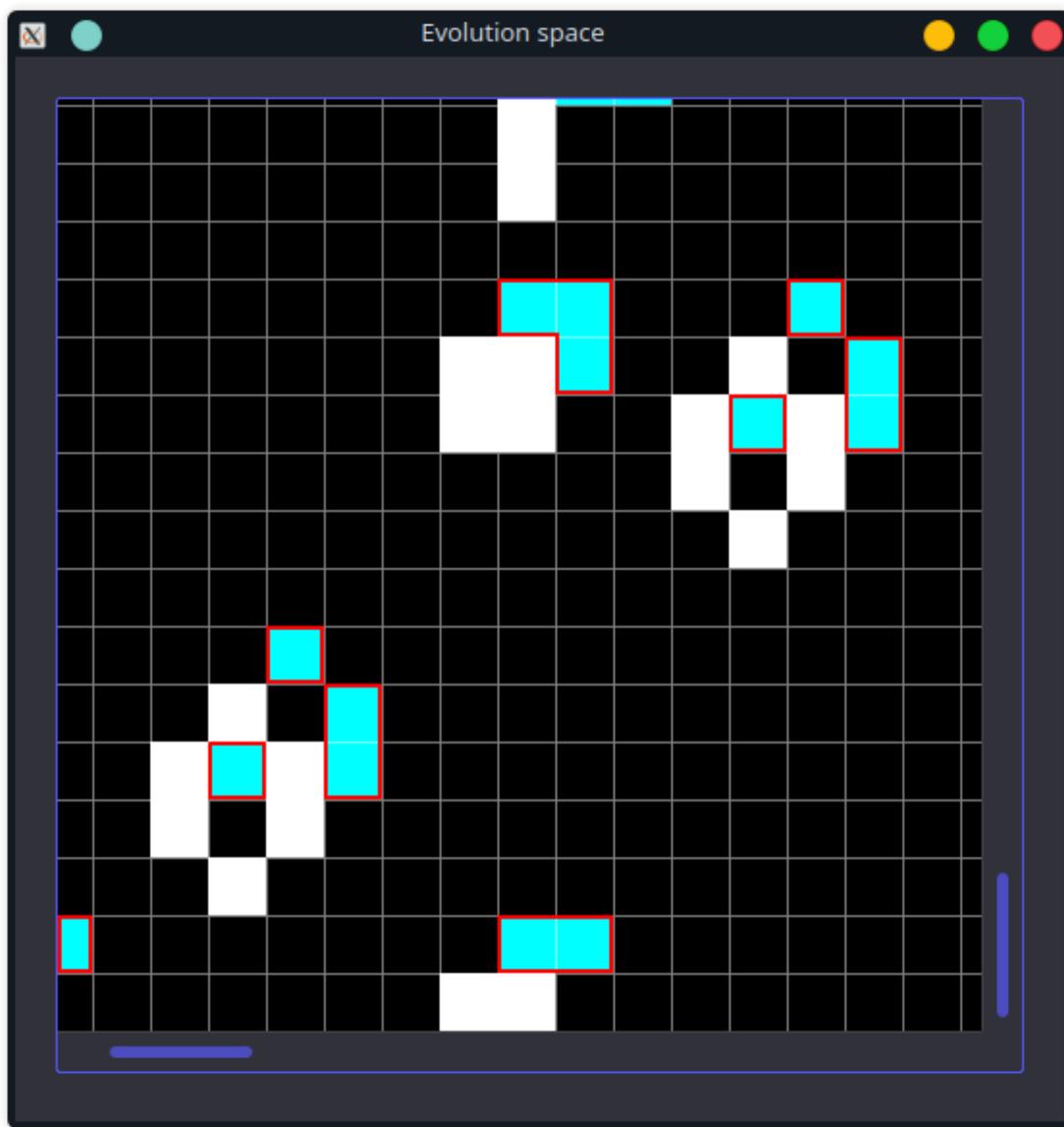


Fig. 6.1.39: IU-CU16-3 Células de proyección en el espacio de evolución del AC

6.1.40. IU-CU17: Cambiar una célula del espacio de evolución del AC

Objetivo: El usuario selecciona una célula del espacio de evolución y su valor es cambiado si el valor era cero este pasa a ser uno y viceversa.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se encuentra de lado izquierdo de la ventana principal (*Settings*) y posteriormente al hacer click en **Draw**

Entradas: Estado de la rejilla

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno



Fig. 6.1.40: IU-CU17 Cambiar una célula del espacio de evolución del AC

6.1.41. IU-CU18: Ventana de configuración

Objetivo: El programa despliega la ventana de configuración al usuario, desde la cual pude manipular cualquier parámetro que deseé en la simulación.

Diseño: Esta pantalla aparece al hacer click en el ícono "vNCASimulator" en la computadora del usuario.

Entradas: Ninguna

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

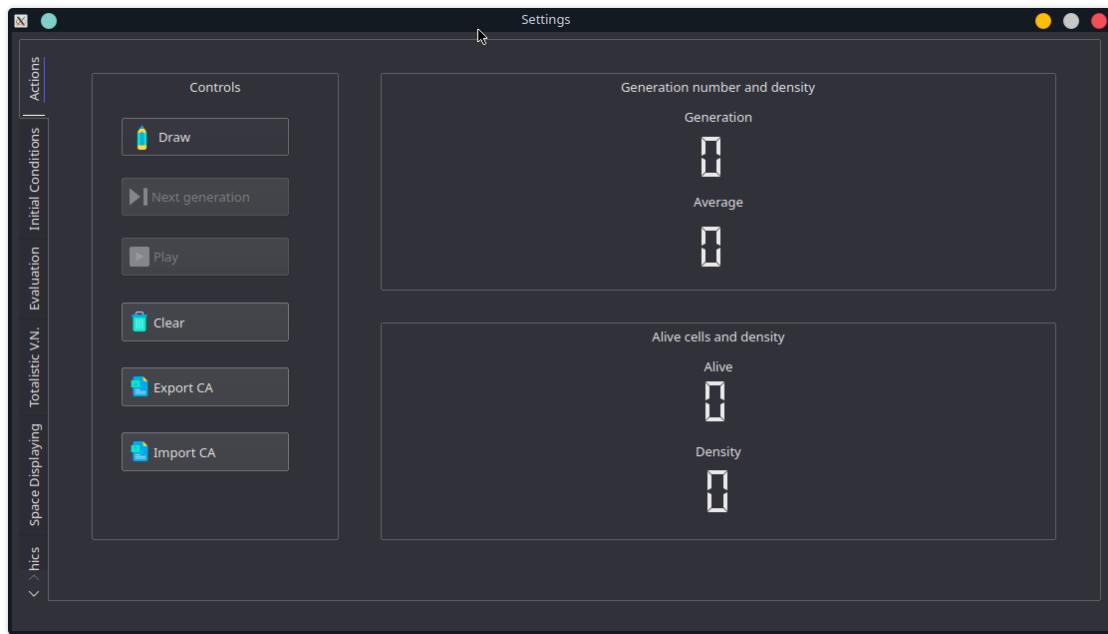


Fig. 6.1.41: IU-CU18 Ventana de configuración

6.1.42. IU-CU19: Desplegar CA con la configuración inicial

Objetivo: El programa obtiene los parámetros necesarios para crear el objeto "Cellular automata"

Diseño: Esta pantalla aparece al hacer click en la pestaña **Space displaying** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Tipo de espacio, configuración inicial, dimensiones del espacio, tamaño de las células.

Salidas: Objeto "Cellular automata"

Comandos: Ninguna **Mensajes:** Ninguno

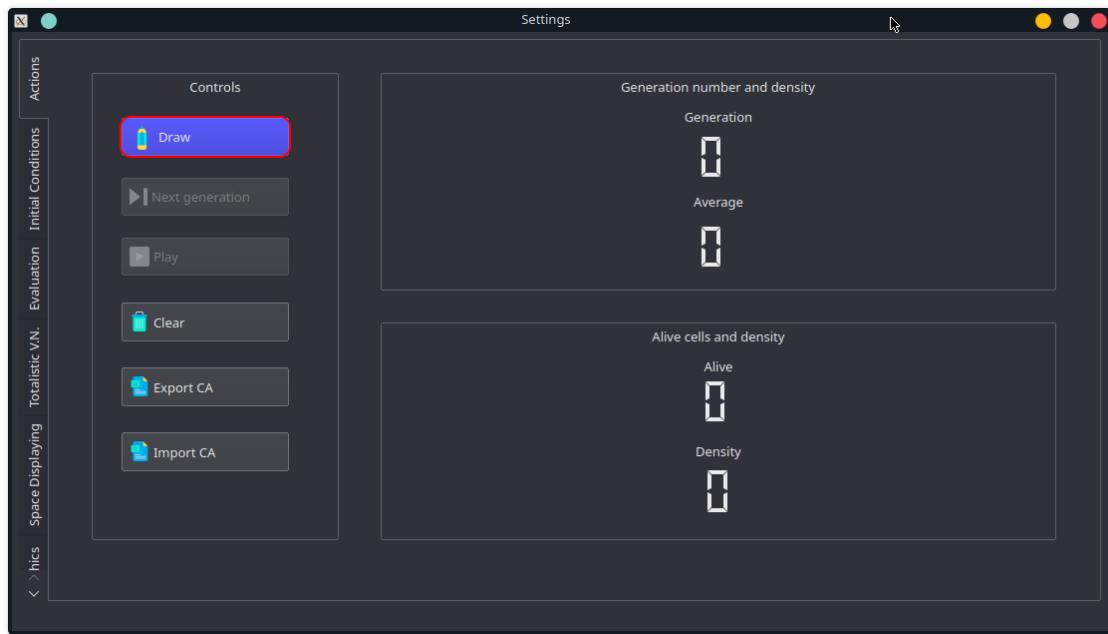


Fig. 6.1.42: IU-CU19 Desplegar CA con la configuración inicial.

6.1.43. IU-CU19.1: Espacio de evolución con la configuración inicial

Objetivo: En el espacio de evolución se pude observar la simulación y resultado de la $t + 1$ generación.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se ecuentra de lado izquierdo de la ventana principal (*Settings*) y posteriormente al hacer click en **Draw**

Entradas: Ninguna

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno



Fig. 6.1.43: IU-CU19.1 Espacio de evolución con la configuración inicial

6.1.44. IU-CU21: Desplegar siguiente generación

Objetivo: Permite calcular la siguiente generación del AC.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Ninguna

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

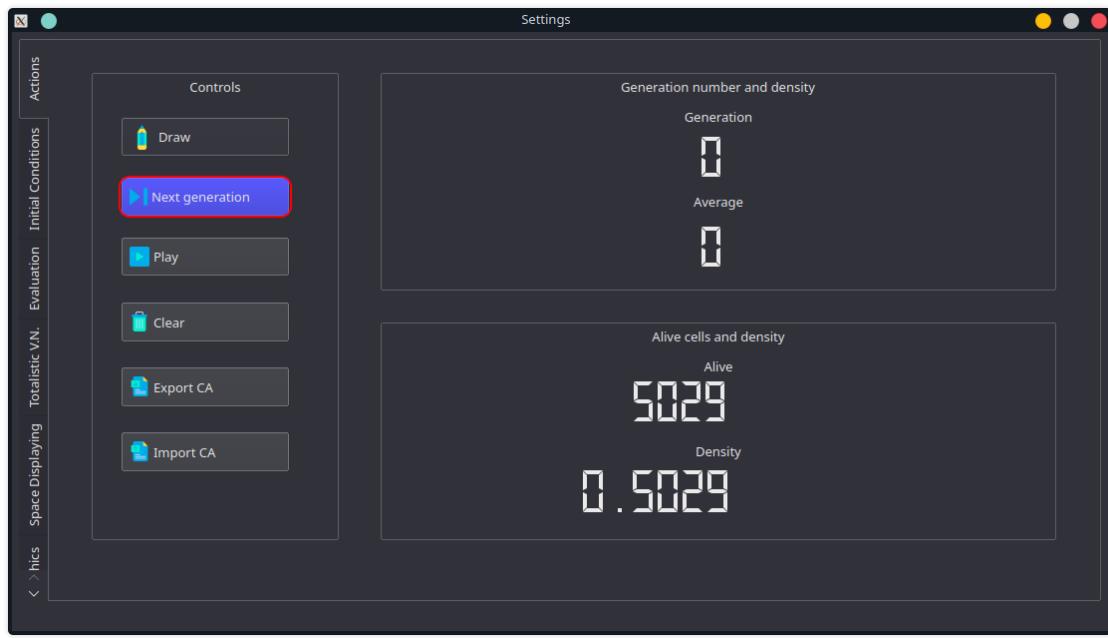


Fig. 6.1.44: IU-CU21 Desplegar siguiente generación

6.1.45. IU-CU21-1: Siguiente generación del AC

Objetivo: Se obtiene la regla de evolución, tipo de espacio y tipo de frontera para calcular la siguiente generación del AC dada una configuración inicial.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Space displaying** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Tipo de espacio, configuración inicial, dimensiones del espacio, tamaño de las células.

Salidas: Objeto "Cellular automata"

Comandos: Ninguna **Mensajes:** Ninguno



Fig. 6.1.45: IU-CU21-1 Siguiente generación del AC

6.1.46. IU-CU26: Graficar población

Objetivo: Al presionar **Next generation** se obtiene la población actual (cantidad de células vivas) y se añade un punto a la gráfica de población.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Población y generación actuales

Salidas: Un punto (x, y)

Comandos: Ninguna **Mensajes:** Ninguno

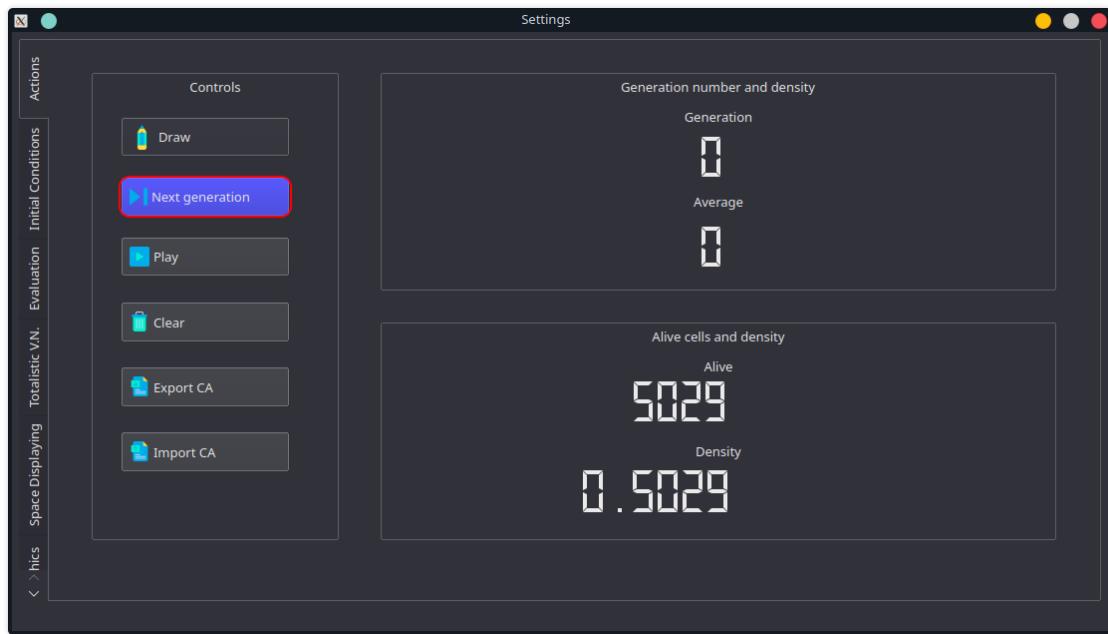


Fig. 6.1.46: IU-CU26 Graficar población

6.1.47. IU-CU26.1: Gráfica de población

Objetivo: Se obtiene la población actual del AC y se añade un punto a la gráfica, dónde el eje x representa la generación y el eje y la cantidad de células vivas en la generación actual.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Graphics** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Un conjunto de puntos

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

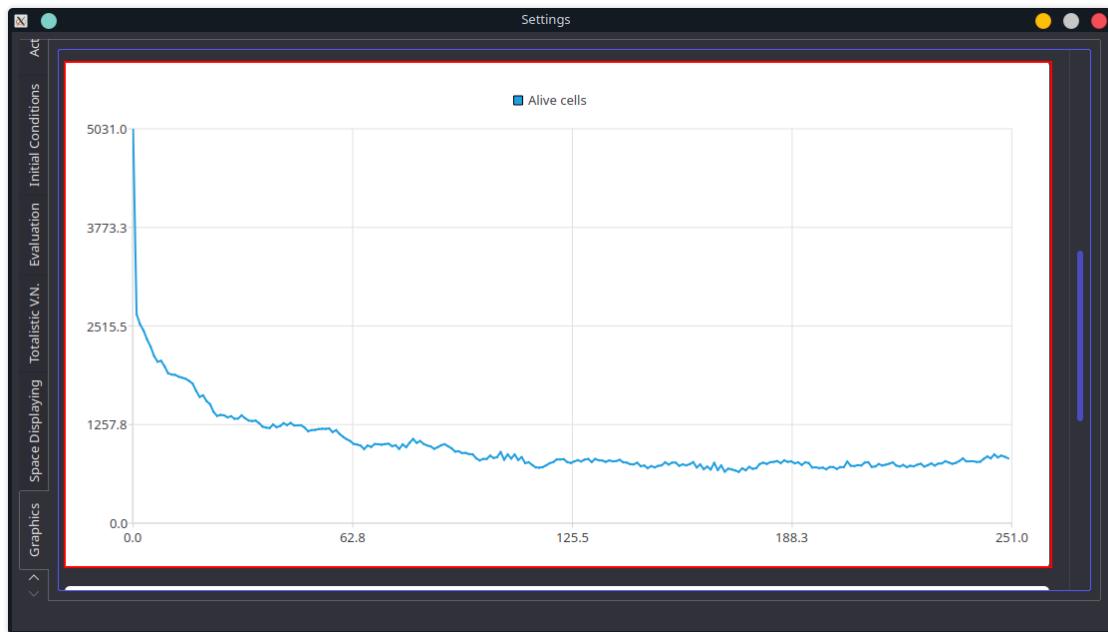


Fig. 6.1.47: IU-CU26.1 Gráfica de población

6.1.48. IU-CU27: Graficar entropía

Objetivo: Al presionar **Next generation** se obtiene la entropía actual y se añade un punto a la gráfica de dentropía.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Entropía y generación actual

Salidas: Un punto (x, y)

Comandos: Ninguna **Mensajes:** Ninguno

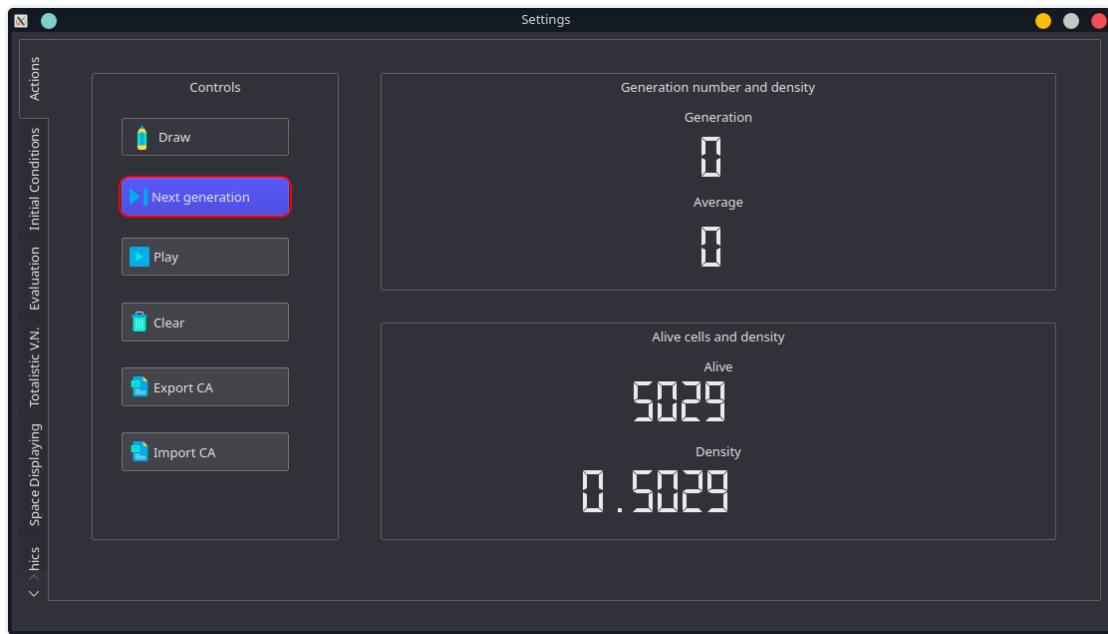


Fig. 6.1.48: IU-CU27 Graficar entropía

6.1.49. IU-CU27.1: Gráfica de entropía

Objetivo: Se obtiene la entropía actual del AC y se añade un punto a la gráfica, dónde el eje x representa la generación y el eje y la entropía en la generación actual.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Graphics** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Un conjunto de puntos (x, y)

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

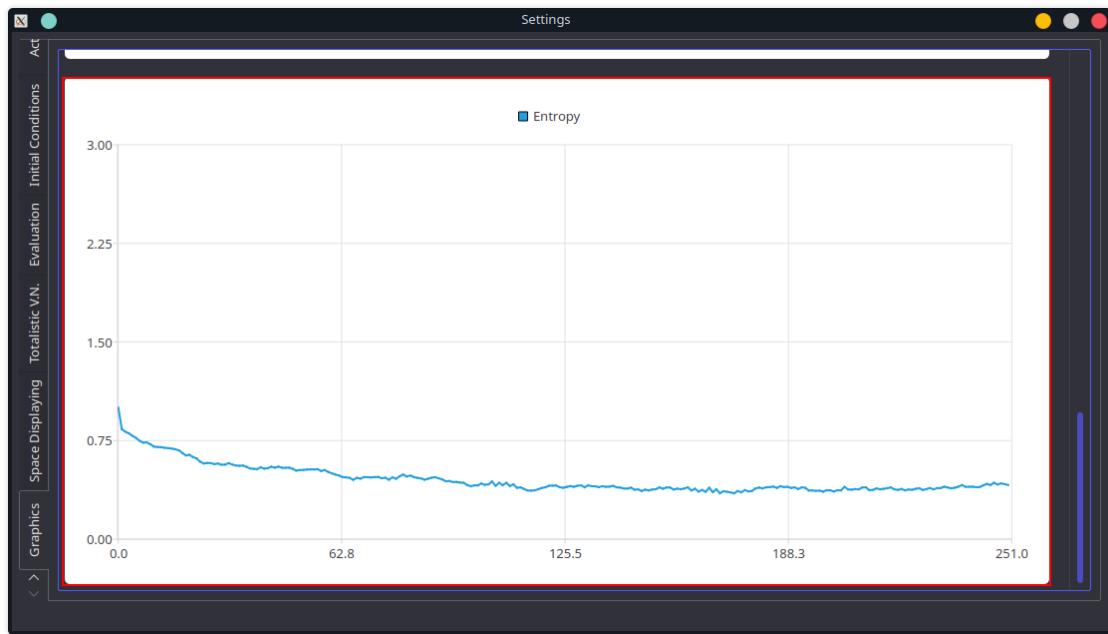


Fig. 6.1.49: IU-CU27.1 Gráfica de entropía

6.1.50. IU-CU28: Graficar polinomio característico

Objetivo: El programa grafica el polinomio característico para la regla de evolución actual.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Space displaying** que se encuentra de lado izquierdo de la ventana principal (*Graphics*)

Entradas: Polinomio característico y un conjunto de puntos

Salidas: Un conjunto de puntos

Comandos: Ninguna **Mensajes:** Ninguno

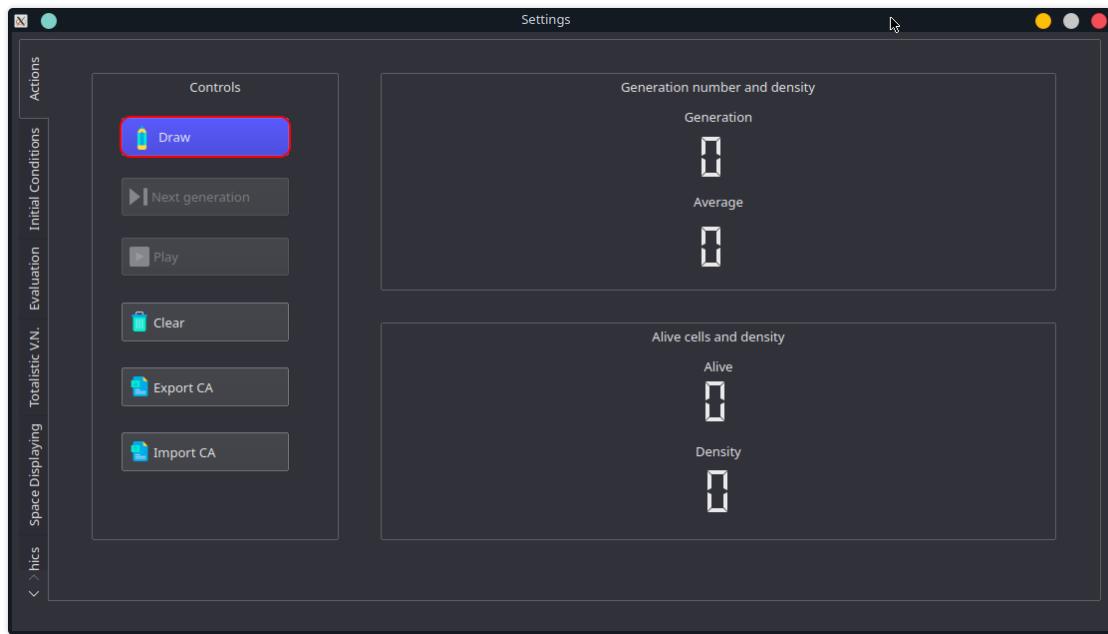


Fig. 6.1.50: IU-CU28 Graficar polinomio característico

6.1.51. IU-CU28.1: Lista de puntos clasificados

Objetivo: El programa muestra una lista con los puntos de intersección entre la recta identidad y el polinomio característico además dichos puntos ya estan clasificados de acuerdo a los diferentes tipos de puntos críticos.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Space displaying** que se encuentra de lado izquierdo de la ventana principal (*Graphics*)

Entradas: Lista de puntos de intersección clasificados.

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

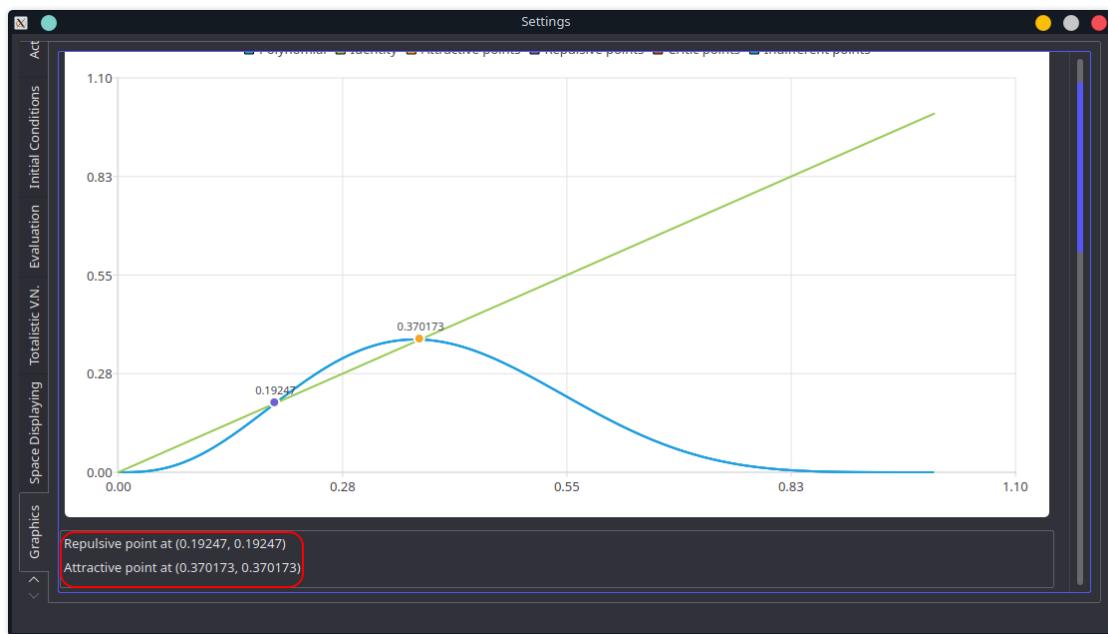


Fig. 6.1.51: IU-CU28.1 Lista de puntos clasificados

6.1.52. IU-CU28.2: Gráfica del polinomio característico

Objetivo: Se grafica el polinomio evaluado en diversos puntos con distancia 1×10^{-3} en un intervalo de $I = [0, 1]$

Diseño: Esta pantalla aparece al hacer click en la pestaña **Graphics** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Un conjunto de puntos

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

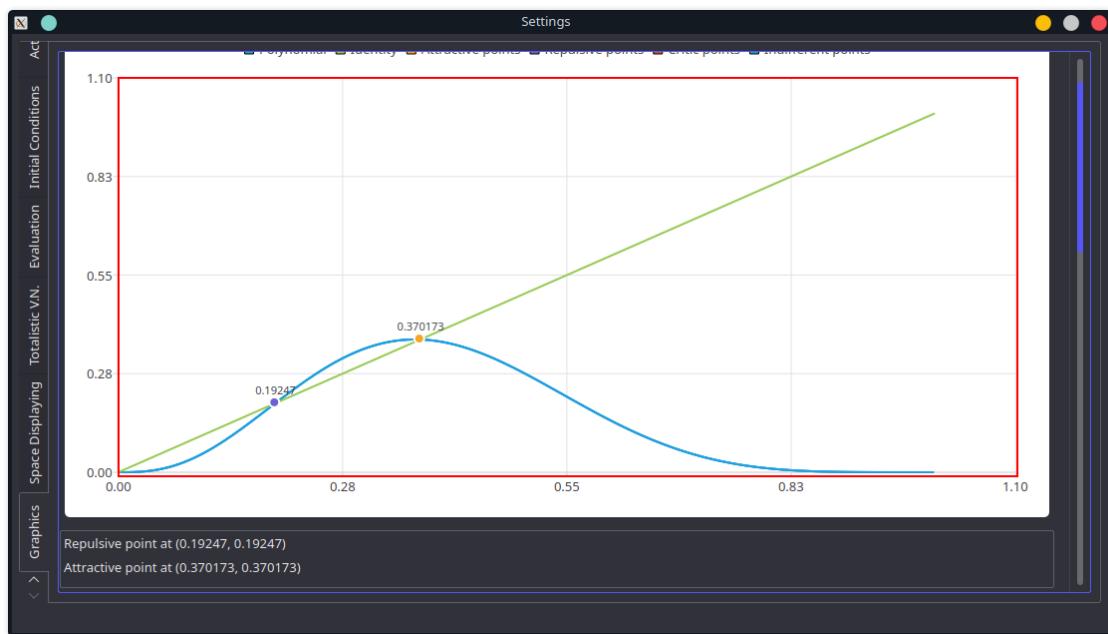


Fig. 6.1.52: IU-CU28.2 Gráfica del polinomio característico

6.1.53. IU-CU29: Graficar recta identidad

Objetivo: El programa añade puntos cuya distancia entre ellos es de 1×10^{-3} y además $x = y$ donde todos los puntos viven en el intervalo $I = [0, 1]$

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Un conjunto de valores $x_n \in R$ donde el intervalo de x_n es $I = [0, 1]$

Salidas: Un vector de puntos (x_n, x_n)

Comandos: Ninguna **Mensajes:** Ninguno

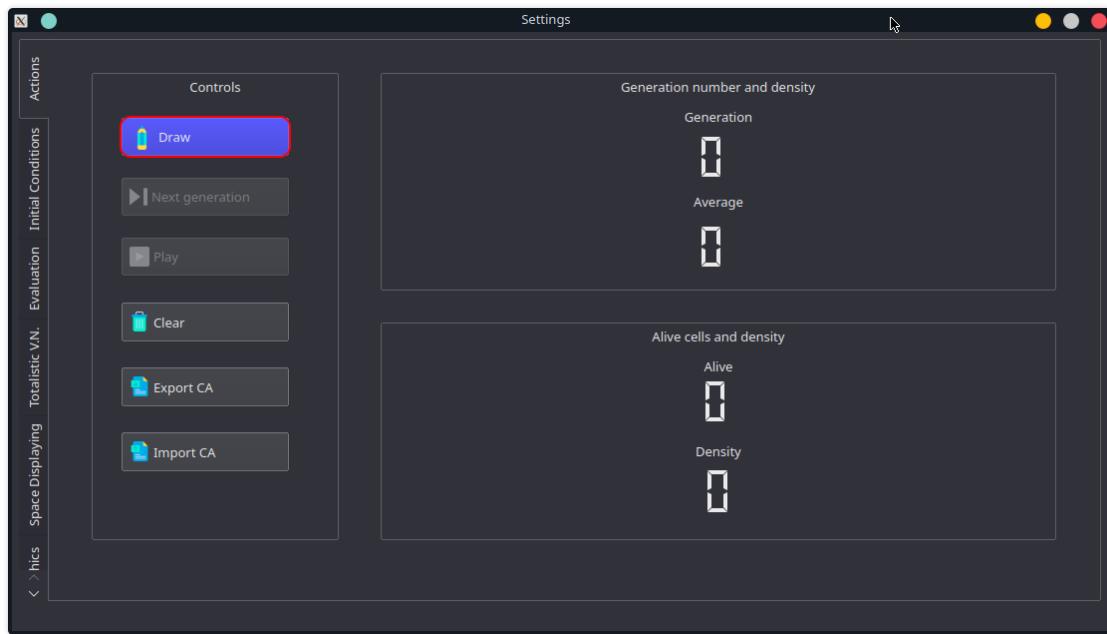


Fig. 6.1.53: IU-CU29 Graficar recta identidad

6.1.54. IU-CU29.1: Gráfica del polinomio característico con recta identidad

Objetivo: Una vez que se han obtenido los puntos de la recta identidad se añaden a la gráfica y son mostrados al usuario.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Space displaying** que se encuentra de lado izquierdo de la ventana principal (*Graphics*)

Entradas: Un conjunto de puntos (x_n, x_n)

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

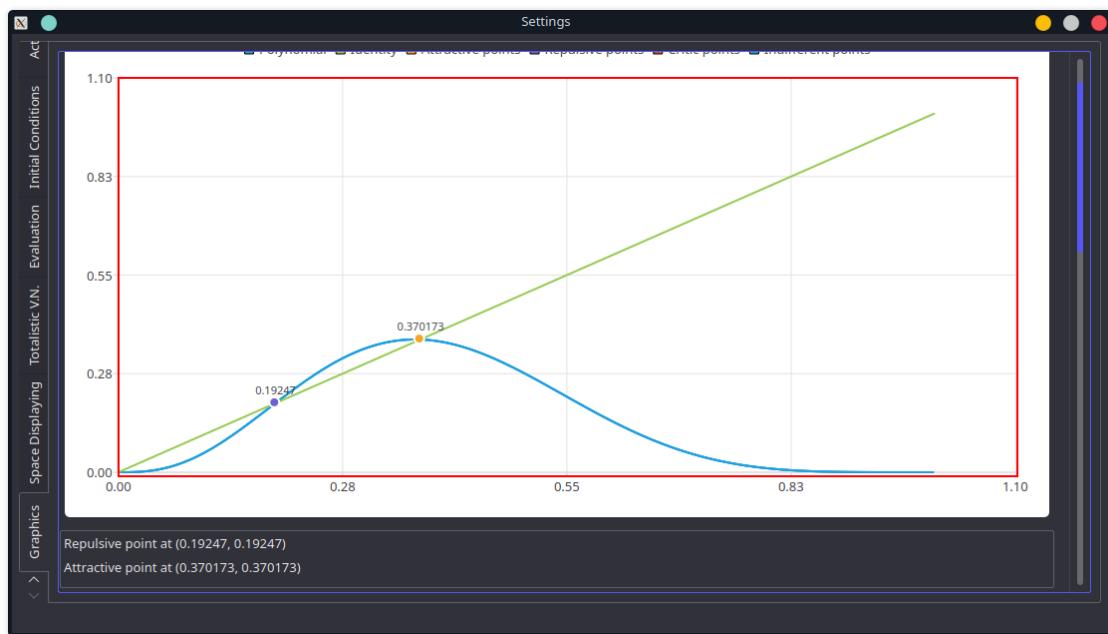


Fig. 6.1.54: IU-CU29.1 Gráfica del polinomio característico con recta identidad

6.1.55. IU-CU30: Graficar puntos de intersección

Objetivo: El programa obtiene los puntos de intersección entre el polinomio característico y la recta identidad.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Un conjunto de valores $x_n \in R$ donde el intervalo de x_n es $I = [0, 1]$

Salidas: Un vector de puntos (x_n, x_n)

Comandos: Ninguna **Mensajes:** Ninguno

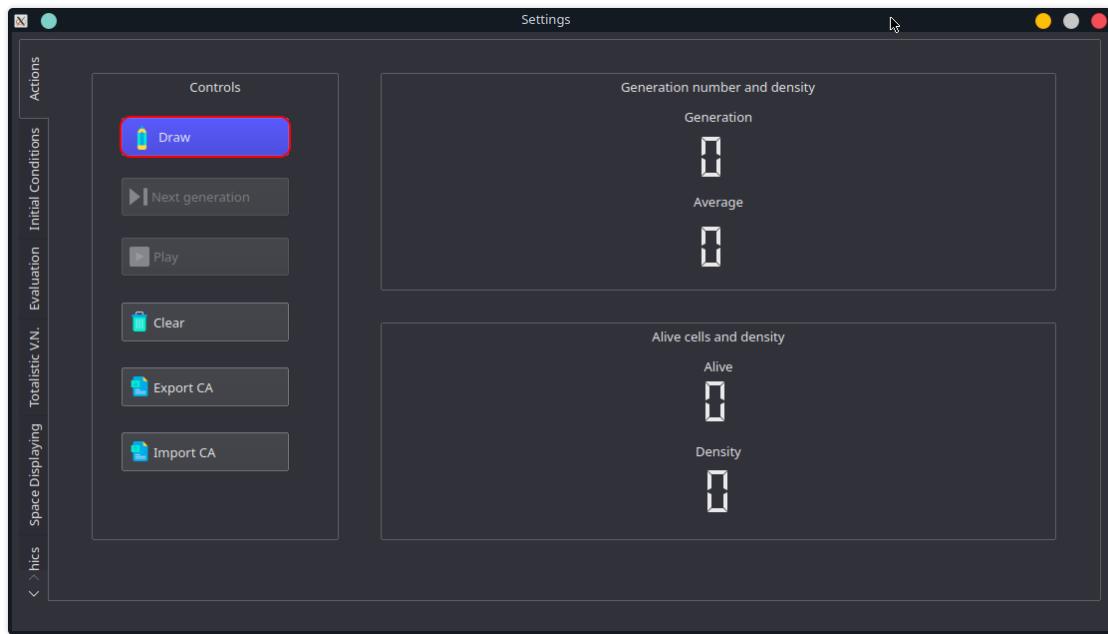


Fig. 6.1.55: IU-CU30 Graficar puntos de intersección

6.1.56. IU-CU30.1: Gráfica del polinomio característico con puntos de intersección del tipo atractivo

Objetivo: Se evalúa $|f'(x)|$ si $|f'(x)| < 1$ el punto se clasifica como atractivo.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Un conjunto de valores $x_n \in R$ donde el intervalo de x_n es $I = [0, 1]$

Salidas: Un vector de puntos (x_n, x_n)

Comandos: Ninguna **Mensajes:** Ninguno

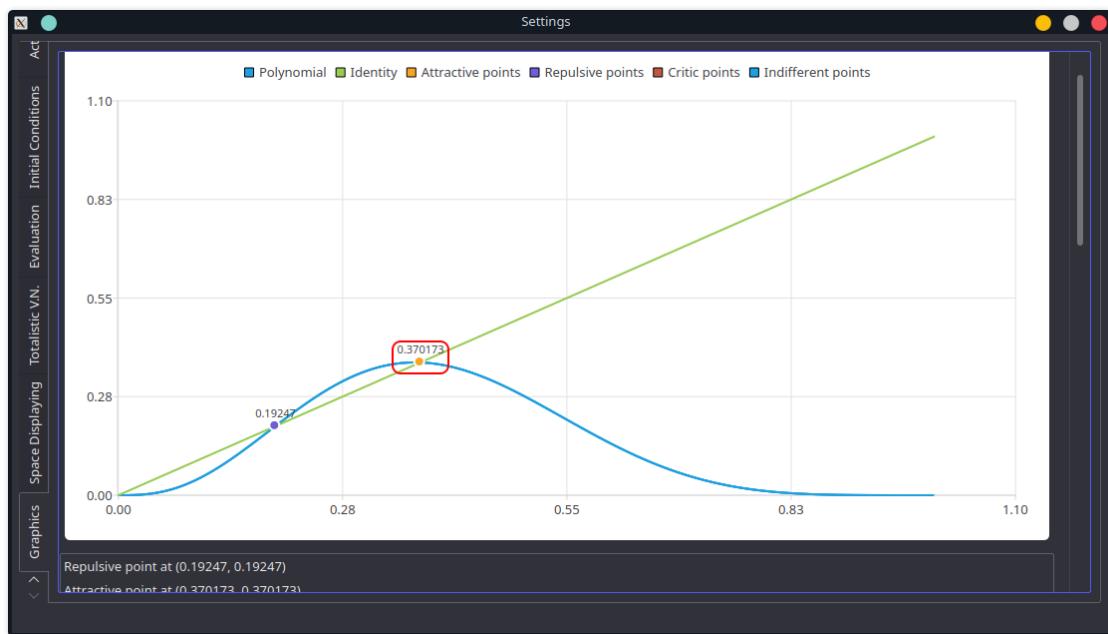


Fig. 6.1.56: IU-CU30.1 Gráfica del polinomio característico con puntos de intersección del tipo atractivo

6.1.57. IU-CU30.2: Gráfica del polinomio característico con puntos de intersección del tipo repulsivo

Objetivo: Se evalúa $|f'(x)|$ si $|f'(x)| > 1$ el punto se clasifica como repulsivo.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Un conjunto de valores $x_n \in R$ donde el intervalo de x_n es $I = [0, 1]$

Salidas: Un vector de puntos (x_n, x_n)

Comandos: Ninguna **Mensajes:** Ninguno



Fig. 6.1.57: IU-CU30.2 Gráfica del polinomio característico con puntos de intersección del tipo repulsivo

6.1.58. IU-CU30.3: Gráfica del polinomio característico con puntos de intersección del tipo crítico

Objetivo: Se evalúa $|f'(x)|$ si $|f'(x)| = 0$ el punto se clasifica como crítico.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Un conjunto de valores $x_n \in R$ donde el intervalo de x_n es $I = [0, 1]$

Salidas: Un vector de puntos (x_n, x_n)

Comandos: Ninguna **Mensajes:** Ninguno

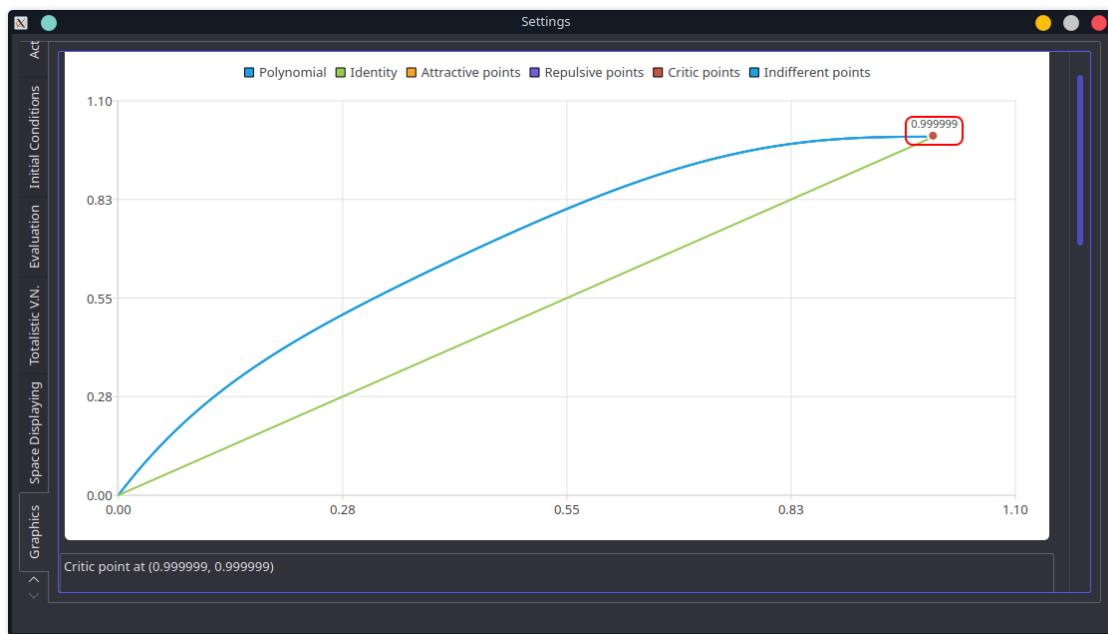


Fig. 6.1.58: IU-CU30.3 Gráfica del polinomio característico con puntos de intersección del tipo crítico

6.1.59. IU-CU31: Iniciar simulación

Objetivo: Al presionar **Play** se inicia el *timer* el cual cada x millisegundos calculará la siguiente generación.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Ninguna

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

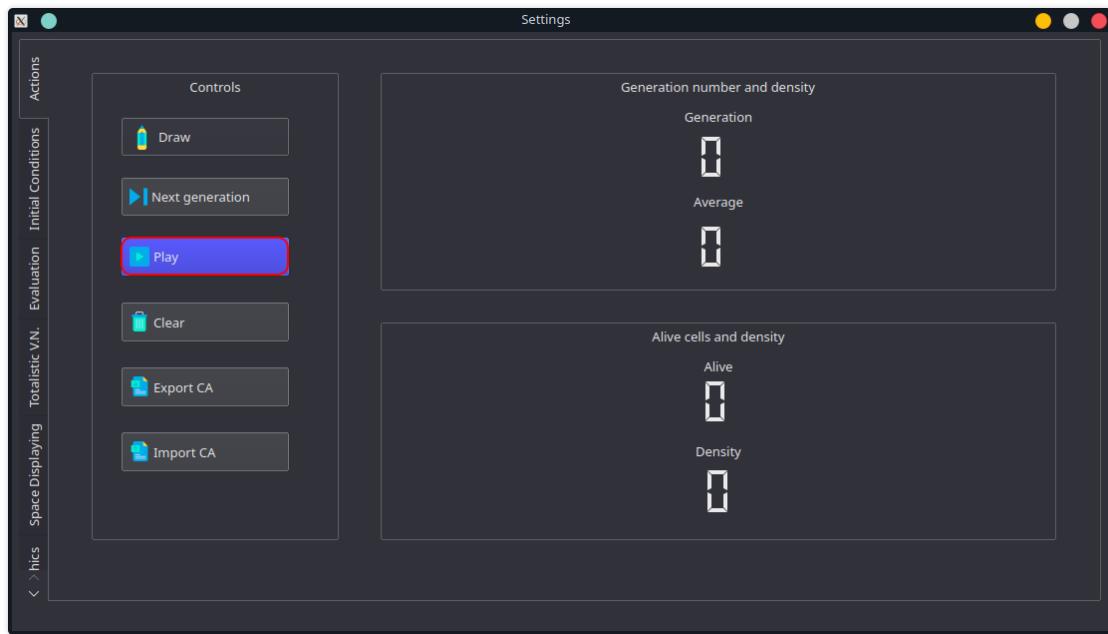


Fig. 6.1.59: IU-CU31 Iniciar simulación

6.1.60. IU-CU32: Detener simulación

Objetivo: Al presionar **Pause** se detiene el *timer* el cual cada x millisegundos calculará la siguiente generación.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Ninguna

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

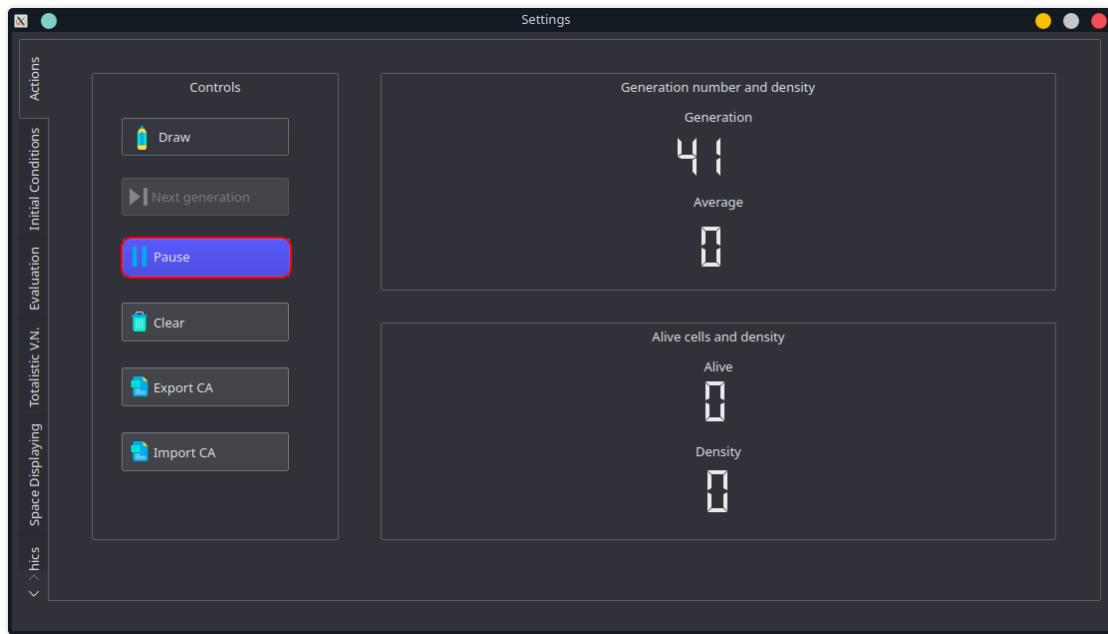


Fig. 6.1.60: IU-CU32 Detener simulación

6.1.61. IU-CU33: Restablecer simulación

Objetivo: Al presionar **Clear** se detiene el *timer* el cual cada x millisegundos calculará la siguiente generación, cada célula del espacio de evoluciones cambia su valor a 0, la población actual se iguala con cero, la generación actual se iguala a cero, los puntos acumulados de la gráfica de población y entropía son eliminados según [IU-CU33.1 Restablecer gráfica de población](#), [IU-CU33.2 Restablecer gráfica de entropía](#), [IU-CU33.3 Restablecer espacio de evoluciones](#).

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Ninguna

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

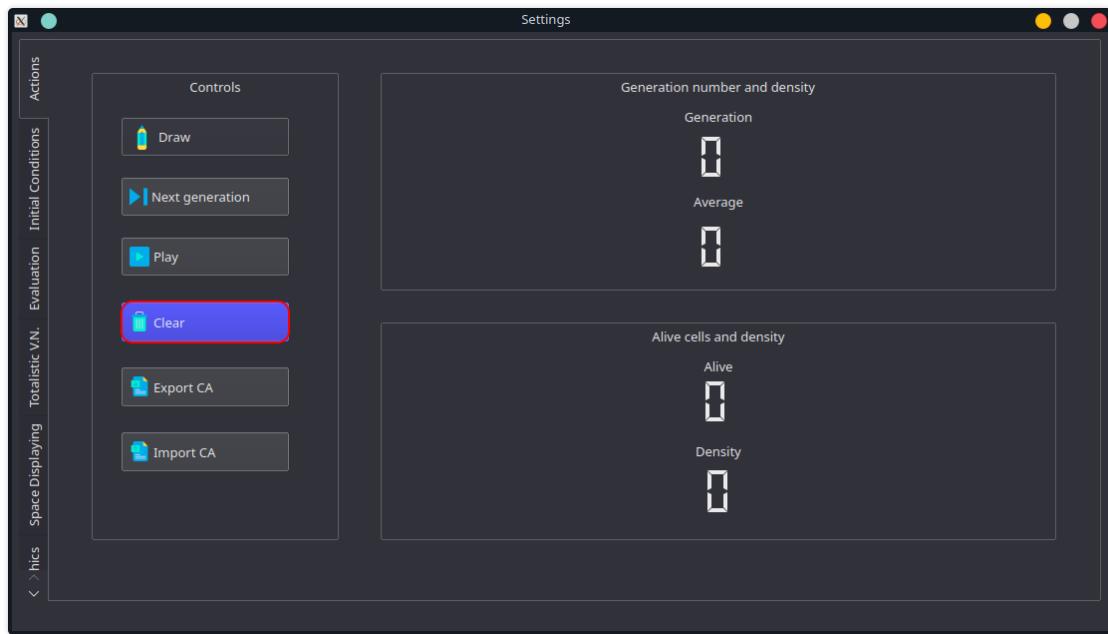


Fig. 6.1.61: IU-CU33 Restablecer simulación

6.1.62. IU-CU33.1: Restablecer gráfica de población

Objetivo: Elimina los puntos acumulados en la gráfica de población.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Graphics** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Ninguna

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

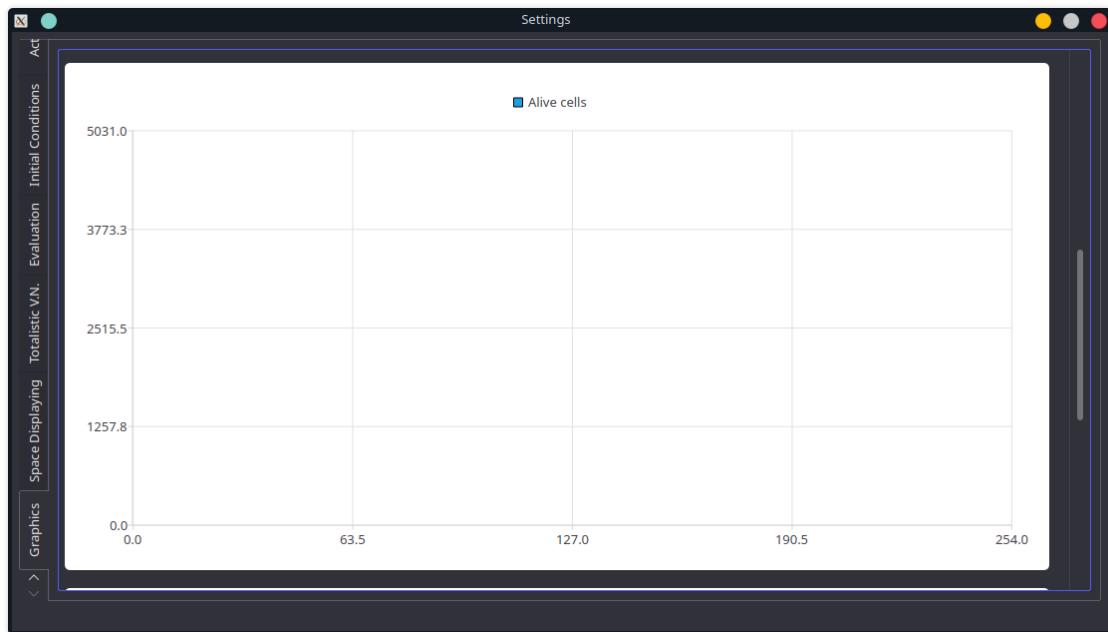


Fig. 6.1.62: IU-CU33.1 Restablecer gráfica de población

6.1.63. IU-CU33.2: Restablecer gráfica de entropía

Objetivo: Elimina los puntos acumulados en la gráfica de entropía.

Diseño: Esta pantalla aparece al hacer click en la pestaña **Graphics** que se encuentra de lado izquierdo de la ventana principal (*Settings*)

Entradas: Ninguna

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno

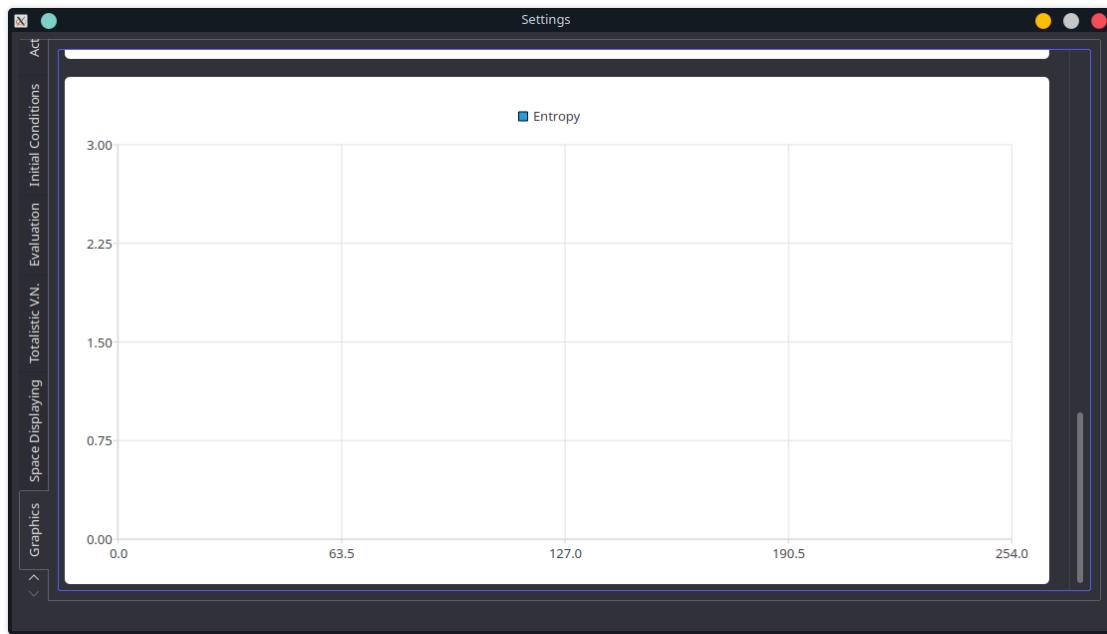


Fig. 6.1.63: IU-CU33.2 Restablecer gráfica de entropía

6.1.64. IU-CU33.3: Restablecer espacio de evolución del AC

Objetivo: Todas las células $c_{i,j}$ del espacio de evolución son igualadas con 0

Diseño: Esta pantalla aparece al hacer click en la pestaña **Actions** que se encuentra de lado izquierdo de la ventana principal (*Settings*) y posteriormente al hacer click en **Draw**

Entradas: Ninguna

Salidas: Ninguna

Comandos: Ninguna **Mensajes:** Ninguno



Fig. 6.1.64: IU-CU33.3 Restablecer espacio de evolución del AC

7

Fase 4: Implementación

7.1. Pruebas unitarias

7.1.1. PU01 Definir la altura del espacio

Descripción completa

Verificar que el valor de la variable altura sea válido, si se ingresa un valor menor a 1 o mayor a 2000 se devuelve el último valor válido ingresado al sistema.

Atributos importantes

Prueba Unitaria:	PU01 Definir la altura del espacio
Versión:	1.0
Caso de uso:	CU01.2 Definir la altura del espacio
Componente:	Pestaña "Initial Conditions"
Precondiciones:	Ninguna
Entradas:	20009
Salida esperada:	2000
Estado:	Correcto
Observaciones:	La verificación del valor límite para la altura es realizada asignando un valor a los atributos <i>minimum</i> y <i>maximum</i> a el objeto <i>QSpinBox</i>
Fecha:	18 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.2. PU02 Definir la anchura del espacio

Descripción completa

Verificar que el valor de la variable anchura sea válido, si se ingresa un valor menor a 1 o mayor a 2000 se devuelve el último valor válido ingresado al sistema.

Atributos importantes

Prueba Unitaria:	PU02 Definir la anchura del espacio
Versión:	1.0
Caso de uso:	CU01.1 Definir la anchura del espacio
Componente:	Pestaña "Initial Conditions"
Precondiciones:	Ninguna
Entradas:	20009
Salida esperada:	2000
Estado:	Correcto
Observaciones:	La verificación del valor límite para la anchura es realizada asignando un valor a los atributos <i>minimum</i> y <i>maximum</i> a los objetos <i>QSpinBox</i>
Fecha:	18 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.3. PU03 Definir vecindad del tipo Moore

Descripción completa

La verificación se realiza revisando el valor bandera del tipo de vecindad si es verdadero es del tipo *Moore* si es falso es del tipo *VonNeumann*

Atributos importantes

Prueba Unitaria:	PU03 Definir vecindad del tipo Moore
Versión:	1.0
Caso de uso:	CU02 Definir vecindad del tipo Moore
Componente:	Pestaña "Evaluation"
Precondiciones:	Ninguna
Entradas:	Verdadero
Salida esperada:	Moore
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	18 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.4. PU04 Definir vecindad del tipo von Neumann

Descripción completa

La verificación se realiza revisando el valor bandera del tipo de vecindad si es verdadero es del tipo *Moore* si es falso es del tipo *VonNeumann*

Atributos importantes

Prueba Unitaria:	PU04 Definir vecindad del tipo von Neumann
Versión:	1.0
Caso de uso:	CU02.1 Definir vecindad del tipo von Neumann
Componente:	Pestaña "Evaluation"
Precondiciones:	Ninguna
Entradas:	Falso
Salida esperada:	von Neumann

Prueba Unitaria:	PU04 Definir vecindad del tipo von Neumann
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	18 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.5. PU05 Definir el tipo de regla como totalista

Descripción completa

La verificación se realiza revisando el valor bandera del tipo de regla, si es verdadero es del tipo *totalista* y si es falso es del tipo *semitotalista*

Atributos importantes

Prueba Unitaria:	PU05 Definir el tipo de regla como totalista
Versión:	1.0
Caso de uso:	CU03 Definir tipo de regla
Componente:	Pestaña "Evaluation"
Precondiciones:	Ninguna
Entradas:	Verdadero
Salida esperada:	totalista
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	18 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.6. PU06 Definir el tipo de regla como semitotalista

Descripción completa

La verificación se realiza revisando el valor bandera del tipo de regla, si es verdadero es del tipo *totalista* y si es falso es del tipo *semi-totalista*

Atributos importantes

Prueba Unitaria:	PU06 Definir el tipo de regla como semitotalista
Versión:	1.0
Caso de uso:	CU03.1 Definir tipo de regla semitotalista
Componente:	Pestaña "Evaluation"
Precondiciones:	Ninguna
Entradas:	Falso
Salida esperada:	semitotalista
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	18 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.7. PU07 Definir los valores de la regla totalista

Descripción completa

La verificación se realiza obteniendo una máscara de bits utilizando un entero sin signo de 32 bits, el número generado utilizando la máscara debe tener una configuración específica la cual representa la regla totalista.

Atributos importantes

Prueba Unitaria:	PU07 Definir los valores de la regla totalista
Versión:	1.0
Caso de uso:	CU04 Definir valores de la regla totalista
Componente:	Pestaña "Evaluation"
Precondiciones:	Ninguna
Entradas:	00000000000000001000101111100111110
Salida esperada:	2096766976
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	18 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.8. PU08 Restablecer valores de la regla totalista

Descripción completa

La verificación se realiza obteniendo una máscara de bits utilizando un entero sin signo de 32 bits, el número generado utilizando la máscara debe ser igual con cero.

Atributos importantes

Prueba Unitaria:	PU08 Restablecer valores de la regla totalista
Versión:	1.0
Caso de uso:	CU05 Definir valores de la regla totalista
Componente:	Pestaña "Totalistic V.N."
Precondiciones:	Ninguna
Entradas:	Ninguna
Salida esperada:	0
Estado:	Correcto
Observaciones:	Se presiona un botón el cual dispara el evento que restablece el valor de la máscara de bits con cero.
Fecha:	18 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.9. PU09 Definir el valor de la variable N_{min}

Descripción completa

Se obtiene el valor de la variable N_{min} y el valor de dicha variable debe estar en el rango de 0 – 8 si la regla es del tipo *Moore* y en el rango de 0 – 4 si es del tipo *Von Neumann*. Si el número ingresado es mayor o menor a alguno de los mencionados previamente el valor almacenado por la variable será el último que fué válido.

Atributos importantes

Prueba Unitaria:	PU09 Definir el valor de la variable N_{min}
Versión:	1.0
Caso de uso:	CU06 Definir los valores de la regla semitotalista
Componente:	Pestaña "Evaluation"
Precondiciones:	Tipo de regla: Moore
Entradas:	80
Salida esperada:	8
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	18 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.10. PU10 Definir el valor de la variable N_{max} **Descripción completa**

Se obtiene el valor de la variable N_{max} y el valor de dicha variable debe estar en el rango de 0 – 8 si la regla es del tipo *Moore* y en el rango de 0 – 4 si es del tipo *Von Neumann*. Si el número ingresado es mayor o menor a alguno de los mencionados previamente el valor almacenado por la variable será el último que fué válido.

Atributos importantes

Prueba Unitaria:	PU10 Definir el valor de la variable N_{max}
Versión:	1.0
Caso de uso:	CU06.1 Definir los valores de la regla semitotalista
Componente:	Pestaña "Evaluation"
Precondiciones:	Tipo de regla: Moore
Entradas:	45
Salida esperada:	4
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	18 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.11. PU11 Definir el valor de la variable S_{min} **Descripción completa**

Se obtiene el valor de la variable S_{min} y el valor de dicha variable debe estar en el rango de 0 – 8 si la regla es del tipo *Moore* y en el rango de 0 – 4 si es del tipo *Von Neumann*. Si el número ingresado es mayor o menor a alguno de los mencionados previamente el valor almacenado por la variable será el último que fué válido.

Atributos importantes

Prueba Unitaria:	PU11 Definir el valor de la variable S_{min}
Versión:	1.0
Caso de uso:	CU06.2 Definir los valores de la regla semitotalista
Componente:	Pestaña "Evaluation"

Prueba Unitaria:	PU11 Definir el valor de la variable S_{min}
Precondiciones:	Tipo de regla: Moore
Entradas:	19
Salida esperada:	1
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	18 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.12. PU08 Definir el valor de la variable S_{max}

Descripción completa

Se obtiene el valor de la variable N_{max} y el valor de dicha variable debe estar en el rango de 0 – 8 si la regla es del tipo *Moore* y en el rango de 0 – 4 si es del tipo *Von Neumann*. Si el número ingresado es mayor o menor a alguno de los mencionados previamente el valor almacenado por la variable será el último que fué válido.

Atributos importantes

Prueba Unitaria:	PU08 Definir el valor de la variable S_{max}
Versión:	1.0
Caso de uso:	CU06.3 Definir los valores de la regla semitotalista
Componente:	Pestaña "Evaluation"
Precondiciones:	Tipo de regla: Moore
Entradas:	90
Salida esperada:	0
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	18 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.13. PU13 Definir el tipo de espacio de evolución como cerrado

Descripción completa

La verificación se realiza revisando el valor bandera de la variable que indica el tipo de espacio de evolución, siendo *falso* el valor para el tipo de espacio abierto y *verdadero* el valor para el espacio del tipo cerrado.

Atributos importantes

Prueba Unitaria:	PU13 Definir el tipo de espacio de evolución como cerrado
Versión:	1.0
Caso de uso:	CU07 Definir tipo de espacio
Componente:	Pestaña "Initial conditions"
Precondiciones:	Ninguna
Entradas:	falso
Salida esperada:	Abierto
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	18 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.14. PU14 Definir el tipo de espacio de evolución como abierto

Descripción completa

La verificación se realiza revisando el valor bandera de la variable que indica el tipo de espacio de evolución, siendo *falso* el valor para el tipo de espacio abierto y *verdadero* el valor para el espacio del tipo cerrado.

Atributos importantes

Prueba Unitaria:		PU14 Definir el tipo de espacio de evolución como abierto
Versión:	1.0	
Caso de uso:	CU07.1 Definir tipo de espacio como abierto	
Componente:	Pestaña "Initial conditions"	
Precondiciones:	Ninguna	
Entradas:	Verdadero	
Salida esperada:	Cerrado	
Estado:	Correcto	
Observaciones:	Ninguna	
Fecha:	18 de noviembre del 2020	
Autor:	Vargas Romero Erick Efraín	

7.1.15. PU15 Establecer configuración inicial del AC de forma aleatoria

Descripción completa

Verificar que el valor de la variable bandera que indica desde donde será inicializado el archivo posee el valor de *verdadero* si la configuración inicial será obtenida desde un archivo y *falso* si la configuración inicial será generada de forma aleatoria.

Atributos importantes

Prueba Unitaria:		PU15 Establecer configuración inicial del AC de forma aleatoria
Versión:	1.0	
Caso de uso:	CU08 Establecer configuración inicial del AC	
Componente:	Pestaña "Initial Conditions"	
Precondiciones:	Ninguna	
Entradas:	Falso	
Salida esperada:	Aleatorio	
Estado:	Correcto	
Observaciones:	Ninguna	
Fecha:	18 de noviembre del 2020	
Autor:	Vargas Romero Erick Efraín	

7.1.16. PU16 Establecer configuración inicial del AC desde archivo

Descripción completa

Verificar que el valor de la variable bandera que indica desde donde será inicializado el archivo posee el valor de *verdadero* si la configuración inicial será obtenida desde un archivo y *falso* si la configuración inicial será generada de forma aleatoria.

Atributos importantes

Prueba Unitaria:	PU16 Establecer configuración inicial del AC desde archivo
Versión:	1.0
Caso de uso:	CU08.1 Establecer configuración inicial desde archivo
Componente:	Pestaña "Initial Conditions"
Precondiciones:	Ninguna
Entradas:	Verdadero
Salida esperada:	Archivo
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	18 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.17. PU17 Exportar objeto CA**Descripción completa**

Se verifica evaluando si existe en memoria un objeto CA, si existe es posible exportar dicho objeto de otra manera es decir si dicho objeto es un apuntador nulo no se exporta nada.

Atributos importantes

Prueba Unitaria:	PU17 Exportar objeto CA
Versión:	1.0
Caso de uso:	CU09 Exportar objeto CA
Componente:	Pestaña "Actions"
Precondiciones:	Ninguna
Entradas:	Dirección del objeto CA
Salida esperada:	Exportar
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	18 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.18. PU18 Definir el tamaño de las células**Descripción completa**

Para graficar las células en el espacio de evoluciones se requiere una variable la cual permite escalar la dimensión de una célula (en pixeles). El tamaño máximo de esta variable es de 50 y el mínimo de 1 si la variable no cumple con lo anterior entonces se coloca el último valor permitido

Atributos importantes

Prueba Unitaria:	PU18 Definir el tamaño de las células
Versión:	1.0
Caso de uso:	CU10 Definir el tamaño de las células
Componente:	Pestaña "Space Displaying"
Precondiciones:	Ninguna

Prueba Unitaria:	PU18 Definir el tamaño de las células
Entradas:	60
Salida esperada:	6
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	18 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.19. PU19 Definir velocidad de evolución

Descripción completa

Verificar que el valor de la variable velocidad de evolución sea válido, si se ingresa un valor menor a 0 o mayor a 5000 millisegundos, se devuelve el último valor válido ingresado al sistema.

Atributos importantes

Prueba Unitaria:	PU19 Definir velocidad de evolución
Versión:	1.0
Caso de uso:	CU10 Definir el tamaño de las células
Componente:	Pestaña "Space Displaying"
Precondiciones:	Ninguna
Entradas:	6000
Salida esperada:	600
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	21 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.20. PU20 Definir escalar de proyección

Descripción completa

Verificar que el valor de la variable escalar de proyección sea válido, si se ingresa un valor menor a 0 o mayor a 100, se devuelve el último valor válido ingresado al sistema.

Atributos importantes

Prueba Unitaria:	PU20 Definir escalar de proyección
Versión:	1.0
Caso de uso:	CU13 Definir el escalar de proyección
Componente:	Pestaña "Space Displaying"
Precondiciones:	Ninguna
Entradas:	1000
Salida esperada:	100
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	21 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.21. PU21 Establecer el estado del gradiente

Descripción completa

Verificar el valor de la variable bandera estado del gradiente, si es *verdadero* el gradiente es mostrado, si es *falso* el gradiente no se muestra.

Atributos importantes

Prueba Unitaria:		PU21 Establecer el estado del gradiente
Versión:	1.0	
Caso de uso:	CU13 Establecer el estado del gradiente	
Componente:	Pestaña "Space Displaying"	
Precondiciones:	Ninguna	
Entradas:	<i>verdadero</i>	
Salida esperada:	Mostrar grid	
Estado:	Correcto	
Observaciones:	Ninguna	
Fecha:	21 de noviembre del 2020	
Autor:	Vargas Romero Erick Efraín	

7.1.22. PU22 Desactivar el gradiente

Descripción completa

Verificar el valor de la variable bandera estado del gradiente, si es *verdadero* el gradiente es mostrado, si es *falso* el gradiente no se muestra.

Atributos importantes

Prueba Unitaria:		PU22 Desactivar el gradiente
Versión:	1.0	
Caso de uso:	CU13.1 Desactivar gradiente	
Componente:	Pestaña "Space Displaying"	
Precondiciones:	Ninguna	
Entradas:	<i>falso</i>	
Salida esperada:	Ocultar grid	
Estado:	Correcto	
Observaciones:	Ninguna	
Fecha:	21 de noviembre del 2020	
Autor:	Vargas Romero Erick Efraín	

7.1.23. PU23 Establecer el estado de la rejilla

Descripción completa

Verificar el valor de la variable bandera estado de la rejilla, si es *verdadero* el valor de dicha bandera es cambiado a *falso* y viceversa

Atributos importantes

Prueba Unitaria:	PU23 Establecer el estado de la rejilla
Versión:	1.0
Caso de uso:	CU14 Establecer estado de la rejilla
Componente:	Pestaña "Space Displaying"
Precondiciones:	Ninguna
Entradas:	<i>falso</i>
Salida esperada:	<i>verdadero</i>
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	21 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.24. PU24 El valor actual de la rejilla es verdadero

Descripción completa

Verificar el valor de la variable bandera estado de la rejilla, si es *verdadero* el valor de dicha bandera es cambiado a *falso* y viceversa

Atributos importantes

Prueba Unitaria:	PU24 El valor actual de la rejilla es verdadero
Versión:	1.0
Caso de uso:	CU14.1 El valor actual de la rejilla es verdadero
Componente:	Pestaña "Space Displaying"
Precondiciones:	Ninguna
Entradas:	<i>verdadero</i>
Salida esperada:	<i>falso</i>
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	21 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.25. PU25 Definir el color de la rejilla

Descripción completa

El color de la rejilla es un valor hexadecimal el cual tiene como valor mínimo 0 y como valor máximo FFFFFF si se ingresa un valor fuera de este rango el sistema toma el último valor válido.

Atributos importantes

Prueba Unitaria:	PU25 Definir el color de la rejilla
Versión:	1.0
Caso de uso:	CU15 Definir el color de la rejilla
Componente:	Pestaña "Space Displaying"
Precondiciones:	Ninguna
Entradas:	<i>FFFFFF</i>
Salida esperada:	<i>FFFFFF</i>
Estado:	Correcto

Prueba Unitaria:	PU25 Definir el color de la rejilla
Observaciones:	Ninguna
Fecha:	21 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.26. PU26 Definir el color de las células

Descripción completa

El color de las células tanto vivas como muertas y de proyección es del tipo hexadecimal el cual tiene como valor mínimo 0 y como valor máximo FFFFFF si se ingresa un valor fuera de este rango el sistema toma el último valor válido.

Atributos importantes

Prueba Unitaria:	PU26 Definir el color de las células
Versión:	1.0
Caso de uso:	CU16 Definir el color de las células
Componente:	Pestaña "Space Displaying"
Precondiciones:	Ninguna
Entradas:	FFFFFF
Salida esperada:	FFFFFF
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	21 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.27. PU27 Definir el color de las células muertas

Descripción completa

El color de las células tanto vivas como muertas y de proyección es del tipo hexadecimal el cual tiene como valor mínimo 0 y como valor máximo FFFFFF si se ingresa un valor fuera de este rango el sistema toma el último valor válido.

Atributos importantes

Prueba Unitaria:	PU27 Definir el color de las células muertas
Versión:	1.0
Caso de uso:	CU16.1 Definir el color de las células muertas
Componente:	Pestaña "Space Displaying"
Precondiciones:	Ninguna
Entradas:	E984AF1
Salida esperada:	E984AF
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	21 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.28. PU28 Definir el color de las células de proyección

Descripción completa

El color de las células tanto vivas como muertas y de proyección es del tipo hexadecimal el cual tiene como valor mínimo 0 y como valor máximo FFFFFF si se ingresa un valor fuera de este rango el sistema toma el último valor válido.

Atributos importantes

Prueba Unitaria:	PU28 Definir el color de las células de proyección
Versión:	1.0
Caso de uso:	CU16.2 Definir el color de las células de proyección
Componente:	Pestaña "Space Displaying"
Precondiciones:	Ninguna
Entradas:	EFAB927
Salida esperada:	EFAB92
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	21 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.29. PU29 Cambiar el estado de una célula

Descripción completa

Las células poseen tres estados 0 representa a una célula muerta, valores positivos representan a una célula viva y finalmente los valores negativos representan a una célula de proyección. Si el valor actual de la célula es mayor a 1 entonces el valor se cambia a cero si es cero se cambia a 1.

Atributos importantes

Prueba Unitaria:	PU29 Cambiar el estado de una célula
Versión:	1.0
Caso de uso:	CU17 Cambiar el estado de una célula
Componente:	Pestaña "Space Displaying"
Precondiciones:	Coordenadas de la célula a cambiar
Entradas:	1
Salida esperada:	0
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	21 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.30. PU30 Desplegar ventana de configuración

Descripción completa

Se obtiene la dirección de un objeto mainwindow si este es igual a un apuntador a nulo no se despliega la ventana de configuración.

Atributos importantes

Prueba Unitaria:	PU30 Desplegar ventana de configuración
Versión:	1.0
Caso de uso:	CU18 Desplegar ventana de configuración
Componente:	Ninguno
Precondiciones:	Ninguna
Entradas:	Dirección del objeto mainwindow
Salida esperada:	Desplegar ventana de configuración
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	21 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.31. PU31 Desplegar el AC con la configuración inicial

Descripción completa

Se obtiene la dirección de un objeto matrixwindow si este es igual a un apuntador a nulo no se despliega la ventana que muestra la configuración inicial del CA.

Atributos importantes

Prueba Unitaria:	PU31 Desplegar el AC con la configuración inicial
Versión:	1.0
Caso de uso:	CU18 Desplegar ventana de configuración
Componente:	Ninguno
Precondiciones:	Ninguna
Entradas:	Dirección del objeto matrixwindow
Salida esperada:	Desplegar ventana con la configuración inicial del AC
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	21 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.32. PU32 Calcular siguiente generación

Descripción completa

Verificar que la siguiente generación del espacio de evolución sea calculada correctamente de acuerdo con las reglas, tipo de vecindad definida y tipo de frontera.

Atributos importantes

Prueba Unitaria:	PU32 Calcular siguiente generación
Versión:	1.0
Caso de uso:	CU20 Calcular siguiente generación
Componente:	Ventana evolution space

Prueba Unitaria:	PU32 Calcular siguiente generación
Precondiciones:	<ul style="list-style-type: none"> ■ Tipo de vecindad: von Neumann ■ Regla: 01000000000000000000000000000000 ■ Tipo de frontera: abierta
Entradas:	010000000
Salida esperada:	000010000
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	29 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.33. PU33 El tipo de regla es semitotalista

Descripción completa

Verificar que la siguiente generación del espacio de evolución sea calculada correctamente de acuerdo con las reglas, tipo de vecindad definida y tipo de frontera.

Atributos importantes

Prueba Unitaria:	PU33 El tipo de regla es semitotalista
Versión:	1.0
Caso de uso:	CU20.1 El tipo de regla es semitotalista
Componente:	Ventana evolution space
Precondiciones:	<ul style="list-style-type: none"> ■ Tipo de vecindad: Moore ■ Regla: S23/B33 ■ Tipo de frontera: abierta
Entradas:	000011011
Salida esperada:	000011011
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	29 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.34. PU34 Calcular densidad de población

Descripción completa

Dado el número de células vivas en la t-ésima generación y las dimensiones del espacio de evolución se puede conocer la densidad de células vivas actual.

Atributos importantes

Prueba Unitaria:	PU34 Calcular densidad de población
Versión:	1.0
Caso de uso:	CU22 Calcular la densidad de población
Componente:	Pestaña actions
Precondiciones:	Ninguna
Entradas:	<ul style="list-style-type: none"> ■ Número de células vivas: 10 ■ Dimensiones del espacio de evolución: 10 x 10
Salida esperada:	0.1
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	29 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.35. PU35 Calcular entropía

Descripción completa

Dada la densidad de células vivas es posible conocer la entropía en la t-ésima generación.

Atributos importantes

Prueba Unitaria:	PU35 Calcular entropía
Versión:	1.0
Caso de uso:	CU23 Calcular entropía
Componente:	Ninguno
Precondiciones:	Ninguna
Entradas:	0.1
Salida esperada:	0.59
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	29 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.36. PU36 Calcular polinomio característico

Descripción completa

Se obtiene el tipo de vecindad y regla de evolución para obtener el polinomio característico mediante teoría del campo promedio.

Atributos importantes

Prueba Unitaria:	PU36 Calcular polinomio característico
Versión:	1.0
Caso de uso:	CU24 Calcular entropía
Componente:	Ninguno

Prueba Unitaria:	PU36 Calcular polinomio característico
Entradas:	<ul style="list-style-type: none"> ■ Tipo de vecindad: Moore ■ Regla de evolución: S23/B33
Salida esperada:	$f(p) = 84p^3q^6 + 56p^4q^5$
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	29 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.37. PU37 Calcular puntos de intersección

Descripción completa

Si se superpone el polinomio característico y la recta identidad se obtienen los puntos de intersección entre ambas funciones, obteniendo como resultado un conjunto de pares ordenados.

Atributos importantes

Prueba Unitaria:	PU37 Calcular puntos de intersección
Versión:	1.0
Caso de uso:	CU25 Calcular puntos de intersección
Componente:	Ninguno
Precondiciones:	Ninguno
Entradas:	<ul style="list-style-type: none"> ■ Vector de puntos del polinomio característico. ■ Vector de puntos de la recta identidad.
Salida esperada:	[(0.19,0.19),(0.37,0.37)]
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	29 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.38. PU38 Graficar puntos de intersección

Descripción completa

Una vez que se han encontrado los puntos de intersección entre la recta identidad y el polinomio característico es posible clasificar dichos puntos haciendo uso de la primer derivada del polinomio característico obtenido. Los puntos son clasificados como atractivos, repulsivos, indiferentes o críticos.

Atributos importantes

Prueba Unitaria:	PU38 Graficar puntos de intersección
Versión:	1.0
Caso de uso:	CU30 Calcular puntos de intersección

Prueba Unitaria:	PU38 Graficar puntos de intersección
Componente:	Ninguno
Precondiciones:	Ninguno
Entradas:	Vector de puntos de intersección entre recta identidad y polinomio característico
Salida esperada:	[(0.37, 0.37)]
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	29 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.39. PU39 Puntos del tipo repulsivo

Descripción completa

Una vez que se han encontrado los puntos de intersección entre la recta identidad y el polinomio característico es posible clasificar dichos puntos haciendo uso de la primer derivada del polinomio característico obtenido. Los puntos son clasificados como atractivos, repulsivos, indiferentes o críticos.

Atributos importantes

Prueba Unitaria:	PU39 Puntos del tipo repulsivo
Versión:	1.0
Caso de uso:	CU30.1 Puntos del tipo repulsivo
Componente:	Ninguno
Precondiciones:	Ninguno
Entradas:	Vector de puntos de intersección entre recta identidad y polinomio característico
Salida esperada:	[(0.19, 0.19)]
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	29 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.40. PU40 Puntos del tipo crítico

Descripción completa

Una vez que se han encontrado los puntos de intersección entre la recta identidad y el polinomio característico es posible clasificar dichos puntos haciendo uso de la primer derivada del polinomio característico obtenido. Los puntos son clasificados como atractivos, repulsivos, indiferentes o críticos.

Atributos importantes

Prueba Unitaria:	PU40 Puntos del tipo crítico
Versión:	1.0
Caso de uso:	CU30.2 Puntos del tipo crítico
Componente:	Ninguno
Precondiciones:	Ninguno
Entradas:	Vector de puntos de intersección entre recta identidad y polinomio característico
Salida esperada:	[]
Estado:	Correcto

Prueba Unitaria:	PU40 Puntos del tipo crítico
Observaciones:	Ninguna
Fecha:	29 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.41. PU41 Puntos del tipo indiferente

Descripción completa

Una vez que se han encontrado los puntos de intersección entre la recta identidad y el polinomio característico es posible clasificar dichos puntos haciendo uso de la primer derivada del polinomio característico obtenido. Los puntos son clasificados como atractivos, repulsivos, indiferentes o críticos.

Atributos importantes

Prueba Unitaria:	PU41 Puntos del tipo indiferente
Versión:	1.0
Caso de uso:	CU30.3 Puntos del tipo indiferente
Componente:	Ninguno
Precondiciones:	Ninguno
Entradas:	Vector de puntos de intersección entre recta identidad y polinomio característico
Salida esperada:	[]
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	30 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.42. PU42 Iniciar simulación

Descripción completa

El valor bandera indica si la simulación está en progreso o no, si el valor es *verdadero* la simulación está siendo ejecutada, si es *falso* la simulación está en pausa.

Atributos importantes

Prueba Unitaria:	PU42 Iniciar simulación
Versión:	1.0
Caso de uso:	CU31 Iniciar simulación
Componente:	Pestaña <i>Actions</i>
Precondiciones:	Ninguno
Entradas:	<i>verdadero</i>
Salida esperada:	En progreso
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	30 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.43. PU43 Detener simulación

Descripción completa

El valor bandera indica si la simulación está en progreso o no, si el valor es *verdadero* la simulación está siendo ejecutada, si es *falso* la simulación está en pausa.

Atributos importantes

Prueba Unitaria:	PU43 Detener simulación
Versión:	1.0
Caso de uso:	CU32 Detener simulación
Componente:	Pestaña <i>Actions</i>
Precondiciones:	Ninguno
Entradas:	<i>falso</i>
Salida esperada:	Sin ejecutar
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	30 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.1.44. PU44 Restablecer simulación

Descripción completa

El valor de las variables, generación actual, densidad actual, promedio de células vivas, número de células vivas actual, configuración del espacio de evolución actual y los vectores que almacenan los valores de población y entropía en tiempo t son reseteados.

Atributos importantes

Prueba Unitaria:	PU44 Restablecer simulación
Versión:	1.0
Caso de uso:	CU33 Restablecer simulación
Componente:	Pestaña <i>Actions</i>
Precondiciones:	Ninguno
Entradas:	<ul style="list-style-type: none"> ■ Generación actual: 5 ■ Densidad actual: 0.16 ■ Promedio acumulado: 0.8 ■ Número de células vivas actual: 4 ■ Configuración: 00000000000000001100011 ■ Población: [(0, 4), (1, 4), (2, 4), (3, 4), (4, 4), (5, 4)] ■ Entropía: [(0, 0.6), (1, 0.6), (2, 0.6), (3, 0.6), (4, 0.6), (5, 0.6)]

Prueba Unitaria:	PU44 Restablecer simulación
Salida esperada:	<ul style="list-style-type: none">■ Generación actual: 0■ Densidad actual: 0■ Promedio acumulado: 0■ Número de células vivas actual: 0■ Configuración: 00000000000000000000000000000000■ Población: []■ Entropía: []
Estado:	Correcto
Observaciones:	Ninguna
Fecha:	30 de noviembre del 2020
Autor:	Vargas Romero Erick Efraín

7.2. Pruebas integradoras



7.2.1. IT01 Configuración

Atributos importantes

Módulo funcional:	IT01 Configuración
Perfiles implicados:	<ul style="list-style-type: none"> ■ Desarrollador ■ Tester
Planificación temporal:	<ol style="list-style-type: none"> 1. Revisión de sintaxis al unir módulos secundarios 2. Revisión de código completo de cada submódulo 3. Definir regla de evolución <ol style="list-style-type: none"> a) Definir regla del tipo semitotalista b) Definir regla del tipo totalista 4. Tipo de frontera <ol style="list-style-type: none"> a) Definir tipo de espacio como abierto b) Definir tipo de espacio como cerrado 5. Cálculo del polinomio característico 6. Cálculo de los puntos de intersección 7. Definición de la configuración inicial del AC <ol style="list-style-type: none"> a) Espacio de evoluciones aleatorio b) Espacio de evoluciones importado dese archivo 8. Definición de la velocidad de evolución 9. Definición de la dimensión del espacio de evoluciones 10. Definición del tamaño de las células 11. Importar archivo

Módulo funcional:	IT01 Configuración
Criterio de verificación:	<p>1. Verificación en las herramientas que ofrece el IDE, que no existan errores de sintaxis</p> <p>2. Verificación de que el código agregado corresponda al módulo funcional y si este requiere cierta modificación para la integración, se conserve la estructura pragmática del módulo</p> <p>3. Integridad del código para la definición de una regla de evolución</p> <ul style="list-style-type: none"> a) Integridad del código para la definición de una regla del tipo semi-totalista b) Integridad del código para la definición de una regla del tipo totalista <p>4. Integridad del código para definir el tipo de frontera</p> <ul style="list-style-type: none"> a) Integridad del código para definir el tipo de espacio como abierto b) Integridad del código para definir el tipo de espacio como cerrado <p>5. Integridad del código para el cálculo el polinomio característico</p> <p>6. Integridad del código para el cálculo de los puntos de intersección</p> <p>7. Integridad del código para la definición de la configuración inicial del AC</p> <ul style="list-style-type: none"> a) Integridad del código para la generación del espacio de evoluciones aleatorio b) Integridad del código para la generación del espacio de evoluciones importado dese archivo <p>8. Integridad del código para la definición de la velocidad de evolución</p> <p>9. Integridad del código para la definición de la dimensión del espacio de evoluciones</p> <p>10. Integridad del código para la definición del tamaño de las células</p> <p>11. Integridad del código para importar archivo</p>

Módulo funcional:	IT01 Configuración
Criterio de aceptación:	<ol style="list-style-type: none"> 1. No existen errores de sintaxis al añadir módulos al programa principal, manteniendo un orden y estructura adecuada. 2. Si un módulo es añadido el código de este se preserva funcionalmente 3. Definir regla de evolución <ol style="list-style-type: none"> a) Definir regla del tipo semitotalista b) Definir regla del tipo totalista 4. Tipo de frontera <ol style="list-style-type: none"> a) Definir tipo de espacio como abierto b) Definir tipo de espacio como cerrado 5. Cálculo del polinomio característico 6. Cálculo de los puntos de intersección 7. Definición de la configuración inicial del AC <ol style="list-style-type: none"> a) Espacio de evoluciones aleatorio b) Espacio de evoluciones importado dese archivo 8. Definición de la velocidad de evolución 9. Definición de la dimensión del espacio de evoluciones 10. Definición del tamaño de las células 11. Importar archivo
Definición de verificaciones:	<ul style="list-style-type: none"> ■ Errores de compilación: Ocurren si la sintaxis del lenguaje es incorrecta
Productos a entregar:	Un objeto del tipo <i>cellular_automata</i> cuyos atributos son los previamente definidos en este módulo.



Caso de prueba TC01

ID del caso de prueba:	TC01
Versión:	1.0
Nombre:	Caso de prueba integrador para el módulo de configuración
Identificador de requerimientos:	<ul style="list-style-type: none"> ■ RF01 ■ RF07 ■ RF09 ■ RF12 ■ RF15 ■ RF16 ■ RF17 ■ RF18 ■ RF19 ■ RF22
Propósito:	Verificar la continua funcionalidad del módulo principal mientras los submódulos correspondientes son añadidos
Dependencias:	N/A
Ambiente de prueba/configuración:	<ul style="list-style-type: none"> ■ Hardware: Equipo de cómputo ■ Software: Compilador de C ++ en su versión 17, IDE y/o editor de texto.
Inicialización:	<ul style="list-style-type: none"> ■ Ratificar que se cuente con cada uno de los requerimientos funcionales que forman parte del módulo de configuración inicial para el simulador. ■ Comprobar que existe un archivo que almacena una configuración inicial específica para una regla dada. ■ Corroborar que las pruebas unitarias hayan arrojado datos correctos. ■ Revisar que se cuente con los datos suficientes para establecer la configuración inicial como la regla, tipo de espacio, tipo de vecindad, etc.
Finalización:	N/A

ID del caso de prueba:	TC01
Acciones:	<ul style="list-style-type: none">■ Se añade cada uno de los submódulos a el módulo principal verificando que el funcionamiento sea correcto
Salida esperada:	Objeto del tipo <i>cellular_automata</i>
Resultado:	Objeto del tipo <i>cellular_automata</i>
Severidad:	Baja
Evidencia:	
Estado:	Correcto



7.2.2. IT02 Evolución

Atributos importantes

Módulo funcional:	IT02 Evolución
Perfiles implicados:	<ul style="list-style-type: none"> ■ Desarrollador ■ Tester
Planificación temporal:	<ol style="list-style-type: none"> 1. Revisión de sintaxis al unir módulos secundarios 2. Revisión de código completo de cada submódulo 3. Cálculo de la siguiente generación 4. Cálculo de la población 5. Cálculo de la densidad 6. Cálculo de la entropía 7. Cálculo de células vivas promedio
Criterio de verificación:	<ol style="list-style-type: none"> 1. Verificación en las herramientas que ofrece el IDE, que no existan errores de sintaxis. 2. Verificación de que el código agregado corresponda al módulo funcional y si este requiere cierta modificación para la integración, se conserve la estructura pragmática del módulo 3. Integridad del código para el cálculo de la siguiente generación 4. Integridad el código para el cálculo de la población 5. Integridad el código para el cálculo de la densidad 6. Integridad el código para el cálculo de la entropía 7. Integridad el código para el cálculo de células vivas promedio

Módulo funcional:	IT02 Evolución
Criterio de aceptación:	<ol style="list-style-type: none">1. No existen errores de sintaxis al añadir módulos al programa principal, manteniendo un orden y estructura adecuada.2. Si un módulo es añadido el código de este se preserva funcionalmente3. Cálculo de la siguiente generación4. Cálculo de la población5. Cálculo de la densidad6. Cálculo de la entropía7. Cálculo de células vivas promedio
Definición de verificaciones:	<ul style="list-style-type: none">■ Errores de compilación: Ocurren si la sintaxis del lenguaje es incorrecta
Productos a entregar:	N/A



Caso de prueba TC02

ID del caso de prueba:	TC02
Versión:	1.0
Nombre:	Caso de prueba integrador para el módulo de evolución
Identificador de requerimientos:	<ul style="list-style-type: none"> ■ RF02 ■ RF03 ■ RF05 ■ RF06 ■ RF14
Propósito:	Verificar la continua funcionalidad del módulo principal mientras los submódulos correspondientes son añadidos
Dependencias:	N/A
Ambiente de prueba-/configuración:	<ul style="list-style-type: none"> ■ Hardware: Equipo de cómputo ■ Software: Compilador de C++ en su versión 17, IDE y/o editor de texto.
Inicialización:	<ul style="list-style-type: none"> ■ Corroborar que los requerimientos que forman parte del módulo hayan pasado el plan de pruebas unitarias satisfactoriamente. ■ Revisar que se cuente con el módulo de configuración funcionando de forma correcta. ■ Establecer una configuración inicial de acuerdo con la regla, tipo de espacio y tipo de vecindad elegidas.
Finalización:	N/A
Acciones:	<ul style="list-style-type: none"> ■ Se añade cada uno de los submódulos a el módulo principal verificando que el funcionamiento sea correcto
Salida esperada:	N/A
Resultado:	N/A
Severidad:	Alta
Evidencia:	
Estado:	Correcto



7.2.3. IT03 Resultado

Atributos importantes

Módulo funcional:	IT03 Resultado
Perfiles implicados:	<ul style="list-style-type: none">■ Desarrollador■ Tester
Planificación temporal:	<ol style="list-style-type: none">1. Revisión de sintaxis al unir módulos secundarios2. Revisión de código completo de cada submódulo3. Desplegar gráfico del polinomio característico4. Desplegar gráfico de población5. Desplegar gráfico de entropía6. Desplegar espacio de evolución<ol style="list-style-type: none">a) Renderizar células vivasb) Renderizar células muertasc) Renderizar células de proyecciónd) Renderizar rejilla7. Exportar archivo

Módulo funcional:	IT03 Resultado
Criterio de verificación:	<ol style="list-style-type: none"> 1. Verificación en las herramientas que ofrece el IDE, que no existan errores de sintaxis 2. Verificación de que el código agregado corresponda al módulo funcional y si este requiere cierta modificación para la integración, se conserve la estructura pragmática del módulo 3. Integridad del código para desplegar gráfico del polinomio característico 4. Integridad del código para desplegar gráfico de población 5. Integridad del código para desplegar gráfico de entropía 6. Integridad del código para desplegar espacio de evolución <ul style="list-style-type: none"> a) Integridad del código para renderizar células vivas b) Integridad del código para renderizar células muertas c) Integridad del código para renderizar células de proyección d) Integridad del código para renderizar rejilla 7. Exportar archivo
Criterio de aceptación:	<ol style="list-style-type: none"> 1. No existen errores de sintaxis al añadir módulos al programa principal, manteniendo un orden y estructura adecuada. 2. Si un módulo es añadido el código de este se preserva funcionalmente 3. Desplegar gráfico del polinomio característico 4. Desplegar gráfico de población 5. Desplegar gráfico de entropía 6. Desplegar espacio de evolución <ul style="list-style-type: none"> a) Renderizar células vivas b) Renderizar células muertas c) Renderizar células de proyección d) Renderizar rejilla 7. Exportar archivo
Definición de verificaciones:	<ul style="list-style-type: none"> ■ Errores de compilación: Ocurren si la sintaxis del lenguaje es incorrecta
Productos a entregar:	



Caso de prueba TC03

ID del caso de prueba:	TC03
Versión:	1.0
Nombre:	Caso de uso integrador para el módulo de resultado
Identificador de requerimientos:	<ul style="list-style-type: none"> ■ RF10 ■ RF11 ■ RF13 ■ RF14 ■ RF20 ■ RF21 ■ RF23 ■ RF24
Propósito:	Verificar la continua funcionalidad del módulo principal mientras los submódulos correspondientes son añadidos
Dependencias:	N/A
Ambiente de prueba-/configuración:	<ul style="list-style-type: none"> ■ Hardware: Equipo de cómputo ■ Software: Compilador de C++ en su versión 17, IDE y/o editor de texto.
Inicialización:	<ul style="list-style-type: none"> ■ Corroborar que los requerimientos que forman parte del módulo hayan pasado el plan de pruebas unitarias satisfactoriamente. ■ Revisar que se cuente con el módulo de configuración funcionando de forma correcta. ■ Revisar que se cuente con el módulo de evolución funcionando de forma correcta. ■ Establecer diferentes colores para cada tipo de célula (vivas, muertas, de proyección), así como para la rejilla.
Finalización:	N/A
Acciones:	<ul style="list-style-type: none"> ■ Se añade cada uno de los submódulos a el módulo principal verificando que el funcionamiento sea correcto

ID del caso de prueba:	TC03
Salida esperada:	
Resultado:	
Severidad:	Media
Evidencia:	
Estado:	Correcto

Parte IV

Exploración y clasificación de reglas

8

Exploración de reglas

En este capítulo se realiza una clasificación de reglas totalistas bajo la vecindad de von Neumann, cuyo propósito es analizar el comportamiento las mismas con diferentes densidades, así como una asociación con sus polinomios característicos obtenidos mediante la utilización de la teoría del campo promedio, la obtención de los puntos que intersectan la recta identidad, así como su clasificación mediante la teoría analizada en el capítulo de sistemas dinámicos. Para realizar esto, se utilizará el simulador desarrollado en este trabajo con la finalidad de agilizar los cálculos, ver el comportamiento en el espacio de evoluciones con diferentes densidades y tamaños de espacio, así como la gráfica y clasificación de los puntos de intersección del polinomio y la recta identidad, visualizar la entropía y las densidades en cada generación.

8.1. Regla 33938756

A continuación se utilizará la regla totalista con el identificador 33938756 para ilustrar la obtención del polinomio característico de campo promedio, además de la gráfica del mismo y clasificación de puntos de intersección del polinomio con la recta identidad.

La regla de evolución está representada como:

$$\varphi(0,0,0,0,0) = 0 \quad \varphi(0,0,0,0,1) = 0 \quad \varphi(0,0,0,1,0) = 1 \quad \varphi(0,0,0,1,1) = 0$$

$$\varphi(0,0,1,0,0) = 0 \quad \varphi(0,0,1,0,1) = 0 \quad \varphi(0,0,1,1,0) = 1 \quad \varphi(0,0,1,1,1) = 0$$

$$\varphi(0,1,0,0,0) = 1 \quad \varphi(0,1,0,0,1) = 0 \quad \varphi(0,1,0,1,0) = 1 \quad \varphi(0,1,0,1,1) = 1$$

$$\varphi(0,1,1,0,0) = 1 \quad \varphi(0,1,1,0,1) = 0 \quad \varphi(0,1,1,1,0) = 1 \quad \varphi(0,1,1,1,1) = 1$$

$$\varphi(1,0,0,0,0) = 1 \quad \varphi(1,0,0,0,1) = 0 \quad \varphi(1,0,0,1,0) = 1 \quad \varphi(1,0,0,1,1) = 0$$

$$\varphi(1,0,1,0,0) = 0 \quad \varphi(1,0,1,0,1) = 0 \quad \varphi(1,0,1,1,0) = 0 \quad \varphi(1,0,1,1,1) = 0$$

$$\varphi(1,1,0,0,0) = 0 \quad \varphi(1,1,0,0,1) = 1 \quad \varphi(1,1,0,1,0) = 0 \quad \varphi(1,1,0,1,1) = 0$$

$$\varphi(1,1,1,0,0) = 0 \quad \varphi(1,1,1,0,1) = 0 \quad \varphi(1,1,1,1,0) = 0 \quad \varphi(1,1,1,1,1) = 0$$

De acuerdo con la ecuación para el cálculo de la probabilidad de un determinado estado en el tiempo

(Ec. 2.6.1) se tiene que:

$$p_{t+1} = (0)p_t^0(1-p_t)^5 + (0)p_t^1(1-p_t)^4 + (1)p_t^1(1-p_t)^4 + (0)p_t^2(1-p_t)^3 + (0)p_t^1(1-p_t)^4 + (0)p_t^2(1-p_t)^3 + (1)p_t^2(1-p_t)^3 + (0)p_t^3(1-p_t)^2 + (1)p_t^1(1-p_t)^4 + (0)p_t^2(1-p_t)^3 + (1)p_t^2(1-p_t)^3 + (1)p_t^3(1-p_t)^2 + (1)p_t^2(1-p_t)^3 + (0)p_t^3(1-p_t)^2 + (1)p_t^3(1-p_t)^2 + (1)p_t^4(1-p_t)^1 + (1)p_t^1(1-p_t)^4 + (0)p_t^2(1-p_t)^3 + (1)p_t^2(1-p_t)^3 + (0)p_t^3(1-p_t)^2 + (0)p_t^2(1-p_t)^3 + (0)p_t^3(1-p_t)^2 + (0)p_t^4(1-p_t)^1 + (0)p_t^4(1-p_t)^1 + (0)p_t^5(1-p_t)^0$$

Simplificando, el polinomio resultante es:

$$3p_t(1-p_t)^4 + 4p_t^2(1-p_t)^3 + 3p_t^3(1-p_t)^2 + p_t^4(1-p_t)$$

Una vez obtenido el polinomio característico se manejará como una función $f(x)$ dentro del intervalo $[0, 1]$ y se construirá una tabla dónde se proponen algunos valores que serán evaluados.

x	$f(x) = 3x(1-x)^4 + 4x^2(1-x)^3 + 3x^3(1-x)^2 + x^4(1-x)$
0.00	0.00*
hline 0.04	0.1077633024
0.08	0.1932064768
0.12	0.2593400832
0.16	0.3088920576
0.20	0.34432
0.24	0.3678234624
0.28	0.3813562368
0.32	0.3866386432
0.36	0.3851698176
0.382	0.382*
0.40	0.37824
0.44	0.3669428224
0.48	0.3521875968
0.52	0.3347116032
0.56	0.3150923776
0.60	0.29376
0.64	0.2710093824
0.68	0.2470125568
0.72	0.2218309632
0.76	0.1954277376
0.80	0.16768
0.84	0.1383911424
0.88	0.1073031168
0.92	0.0741087232
0.96	0.0384638976
1.00	0.00

Con la información anterior es posible graficar el polinomio característico y la recta identidad con el fin de localizar los puntos de intersección entre ambas funciones, tal y como se muestra en Fig. 8.1.1.

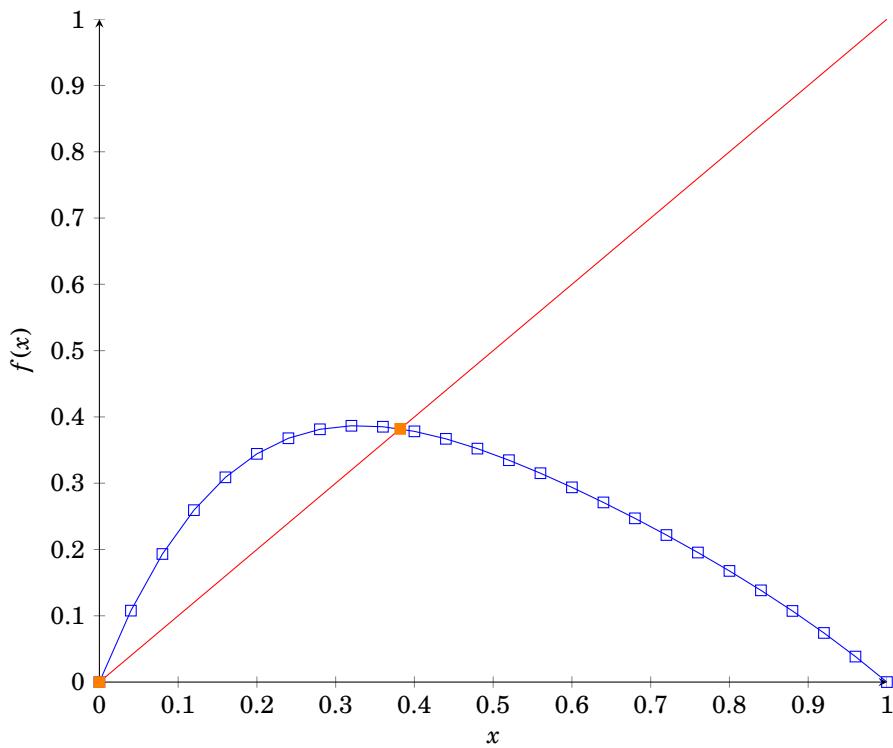
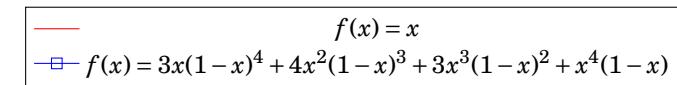


Fig. 8.1.1: Gráfica del polinomio correspondiente a la regla 33938756

Obtenida la gráfica, se puede notar que existen dos puntos de intersección entre el polinomio característico y la recta identidad. Estos puntos se ubican en $(0,0)$ y $(0.382, 0.382)$ correspondientemente.

Finalmente, para proporcionar una clasificación a los puntos de intersección se debe hallar la derivada del polinomio característico, evaluar el n -ésimo punto de intersección en la derivada y de acuerdo al criterio de sistemas dinámicos, asignar la clasificación adecuada para el n -ésimo punto de intersección.

El criterio de sistemas dinámicos indica que:

Sea x^* un punto fijo:

Si $|f'(x^*)| < 1$ entonces x^* es del tipo atractivo

Si $|f'(x^*)| > 1$ entonces x^* es del tipo repulsivo

Si $|f'(x^*)| = 1$ entonces x^* es del tipo indiferente

Si existe un n que satisface $f^n(x^*) = x^*$ entonces x^* es del tipo cíclico

Calculado la derivada del polinomio característico, se tiene:

$$f'(x) = 5x^4 - 20x^3 + 27x^2 - 16x + 3$$

Evaluando los puntos de intersección:

$$f'(0) = 5(0)^4 - 20(0)^3 + 27(0)^2 - 16(0) + 3 = 3$$

$$f'(0.382) = 5(0.382)^4 - 20(0.382)^3 + 27(0.382)^2 - 16(0.382) + 3 = -0.1804$$

De acuerdo al criterio de sistemas dinámicos, el punto ubicado en $(0,0)$ indica ser un punto repulsivo como consecuencia por ser mayor que 1 al ser evaluado en la derivada del polinomio, finalmente, el punto ubicado en $(0.382, 0.382)$ indica ser un punto del tipo atractivo por ser menor que 1 al ser evaluado en la derivada del polinomio.

9

Reglas exploradas

9.1. Asociación de reglas y polinomios

En la tabla que se muestra a continuación se pueden apreciar las reglas del tipo totalistas exploradas bajo la vecindad de von Neumann.

Se utilizó el simulador desarrollado previamente con la finalidad de obtener el polinomio característico, así como el tipo de puntos fijos, que son los que tienen una intersección con la recta identidad. Además se incluye la gráfica del polinomio con la recta identidad con fines ilustrativos para que se pueda apreciar mejor el comportamiento de la curva. Cabe destacar que el identificador es la representación decimal de cada regla explorada.

Identificador	Polinomio	Puntos	Gráfica
539312132	$2pq^4 + p^2q^3 + 2p^3q^2 + p^4q$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.19, 0.19): Tipo atractivo. 	
2436109	$q^5 + 2pq^4 + 3p^2q^3 + 3p^3q^2$	<ul style="list-style-type: none"> ■ (0.37, 0.37): Tipo atractivo. 	
1853030134	$3pq^4 + 7p^2q^3 + 8p^3q^2 + 4p^4q$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.66, 0.66): Tipo atractivo. 	
4189052334	$4pq^4 + 7p^2q^3 + 6p^3q^2 + 5p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.99, 0.99): Tipo atractivo. 	

Identificador	Polinomio	Puntos	Gráfica
2111306941	$q^5 + 3pq^4 + 8p^2q^3 + 7p^3q^2 + 5p^4q$	<ul style="list-style-type: none"> ■ (0.70, 0.70): Tipo atractivo. 	<p>A graph on a coordinate plane where both axes range from 0 to 1. A blue curve starts at (0,0), rises to a local maximum at approximately (0.7, 0.7), and then descends back towards (1,0). A straight red line represents the identity function $y = x$.</p>
1077936134	$2pq^4 + p^3q^2 + p^4q$	<ul style="list-style-type: none"> ■ (0,0): Tipo repulsivo. ■ (0.16, 0.16): Tipo atractivo. 	<p>A graph on a coordinate plane where both axes range from 0 to 1. A blue curve starts at (0,0), rises to a local maximum at approximately (0.16, 0.16), and then descends back towards (1,0). A straight red line represents the identity function $y = x$.</p>
8454144	$pq^4 + p^4q$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo atractivo. 	<p>A graph on a coordinate plane where both axes range from 0 to 1. A blue curve starts at (0,0), rises to a local maximum at approximately (0.0, 0.0), and then descends back towards (1,0). A straight red line represents the identity function $y = x$.</p>
50627360	$pq^4 + 4p^2q^3 + p^3q^2 + p^4q$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo atractivo. 	<p>A graph on a coordinate plane where both axes range from 0 to 1. A blue curve starts at (0,0), rises to a local maximum at approximately (0.0, 0.0), and then descends back towards (1,0). A straight red line represents the identity function $y = x$.</p>

Identificador	Polinomio	Puntos	Gráfica
1884435473	$q^5 + pq^4 + 4p^2q^3 + 3p^3q^2 + 2p^4q$	<ul style="list-style-type: none"> ■ (0.37, 0.37): Tipo atractivo. 	
3943755596	$2pq^4 + 6p^2q^3 + 4p^3q^2 + 4p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.57, 0.57): Tipo atractivo. ■ (0.99, 0.99): Tipo repulsivo. 	
2422207638	$4pq^4 + 5p^2q^3 + 6p^3q^2 + p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.52, 0.52): Tipo atractivo. ■ (1, 1): Tipo repulsivo. 	
177146253	$q^5 + 3pq^4 + 3p^2q^3 + 4p^3q^2 + 2p^4q$	<ul style="list-style-type: none"> ■ (0.43, 0.43): Tipo atractivo. 	

Identificador	Polinomio	Puntos	Gráfica
2585170240	$pq^4 + 5p^2q^3 + 2p^3q^2 + 2p^4q + p^5$	<ul style="list-style-type: none"> ■ (0,0,0,0): Tipo repulsivo. ■ (0.18,0.18): Tipo atractivo. ■ (1,1): Tipo repulsivo. 	
1686636573	$q^5 + 2pq^4 + p^2q^3 + 3p^3q^2 + 3p^4q$	<ul style="list-style-type: none"> ■ (0.35,0.35): Tipo atractivo. 	
1540729301	$q^5 + 4pq^4 + 4p^2q^3 + 6p^3q^2 + 4p^4q$	<ul style="list-style-type: none"> ■ (0.58,0.58): Tipo atractivo. 	
2737841485	$q^5 + 2pq^4 + 5p^2q^3 + 3p^3q^2 + p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.43,0.43): Tipo atractivo. ■ (1,1): Tipo repulsivo. 	

Identificador	Polinomio	Puntos	Gráfica
457382132	$3pq^4 + 5p^2q^3 + 5p^3q^2 + p^4q$	<ul style="list-style-type: none"> ■ (0,0,0): Tipo repulsivo. ■ (0.45,0.45): Tipo atractivo. 	
1457500966	$4pq^4 + 6p^2q^3 + 6p^3q^2 + 3p^4q$	<ul style="list-style-type: none"> ■ (0,0,0): Tipo repulsivo. ■ (0.57,0.57): Tipo atractivo. 	
3996909215	$q^5 + 4pq^4 + 6p^2q^3 + 8p^3q^2 + 4p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.99,0.99): Tipo atractivo. 	
4285263798	$5pq^4 + 5p^2q^3 + 10p^3q^2 + 4p^4q + p^5$	<ul style="list-style-type: none"> ■ (0,0,0): Tipo repulsivo. ■ (0.99,0.99): Tipo atractivo. 	

Identificador	Polinomio	Puntos	Gráfica
4223663	$q^5 + 2pq^4 + 4p^2q^3 + 4p^3q^2$	■ (0.40, 0.40): Tipo atractivo.	
3085644	$3pq^4 + 6p^2q^3 + 2p^3q^2$	■ (0.0, 0.0): Tipo repulsivo. ■ (0.40, 0.40): Tipo atractivo.	
2049698883	$q^5 + 2pq^4 + 3p^2q^3 + 7p^3q^2 + 4p^4q$	■ (0.54, 0.54): Tipo atractivo.	
341699601	$q^5 + 2pq^4 + 3p^2q^3 + 7p^3q^2 + p^4q$	■ (0.44, 0.44): Tipo atractivo.	

Identificador	Polinomio	Puntos	Gráfica
871013027	$q^5 + pq^4 + 5p^2q^3 + 7p^3q^2 + 3p^4q$	<ul style="list-style-type: none"> ■ (0.53, 0.53): Tipo atractivo. 	
2147483649	$q^5 + p^5$	<ul style="list-style-type: none"> ■ (0.24, 0.24): Tipo repulsivo. ■ (1.0, 1.0): Tipo repulsivo. 	
3480215414	$5pq^4 + 8p^2q^3 + 7p^3q^2 + 3p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.74, 0.74): Tipo atractivo. ■ (0.99, 0.99): Tipo repulsivo. 	
2422207	$q^5 + 4pq^4 + 5p^2q^3 + 4p^3q^2 + p^4q$	<ul style="list-style-type: none"> ■ (0.48, 0.48): Tipo atractivo. 	

Identificador	Polinomio	Puntos	Gráfica
4143971324	$4pq^4 + 8p^2q^3 + 10p^3q^2 + 4p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.99, 0.99): Tipo atractivo. 	
1867275334	$2pq^4 + 5p^2q^3 + 6p^3q^2 + 3p^4q$	<ul style="list-style-type: none"> ■ (0.0, 0.00): Tipo repulsivo. ■ (0.5, 0.5): Tipo atractivo. 	
309203110	$2pq^4 + 4p^2q^3 + 6p^3q^2$	<ul style="list-style-type: none"> ■ (0.06, 0.06): Tipo repulsivo. ■ (0.37, 0.37): Tipo atractivo. 	
2702504454	$2pq^4 + 5p^2q^3 + 2p^3q^2 + 2p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.06, 0.06): Tipo repulsivo. ■ (0.35, 0.35): Tipo atractivo. ■ (1.0, 1.0): Tipo repulsivo. 	

Identificador	Polinomio	Puntos	Gráfica
1152074856	$pq^4 + 5p^2q^3 + 4p^3q^2 + 2p^4q$	<ul style="list-style-type: none"> ■ (0,0,0): Tipo repulsivo. ■ (0.26,0.26): Tipo atractivo. 	
3047921227	$q^5 + 2pq^4 + 7p^2q^3 + 4p^3q^2 + 3p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.56,0.56): Tipo atractivo. ■ (0.99,0.99): Tipo repulsivo. 	
687883884	$pq^4 + 6p^2q^3 + p^3q^2 + 2p^4q$	<ul style="list-style-type: none"> ■ (0,0,0): Tipo repulsivo. ■ (0.25,0.25): Tipo atractivo. 	
1103116065	$q^5 + pq^4 + 5p^2q^3 + 2p^3q^2 + 2p^4q$	<ul style="list-style-type: none"> ■ (0.38,0.38): Tipo atractivo. 	

Identificador	Polinomio	Puntos	Gráfica
33938756	$3pq^4 + 4p^2q^3 + 3p^3q^2 + p^4q$	<ul style="list-style-type: none"> ■ (0,0,0,0): Tipo repulsivo. ■ (0.38,0.38): Tipo atractivo. 	
2149679104	$pq^4 + p^3q^2 + p^4q + p^5$	<ul style="list-style-type: none"> ■ (1,0,1,0): Tipo repulsivo. ■ (0,0,0,0): Tipo atractivo. 	
1745944769	$q^5 + pq^4 + 2p^2q^3 + p^3q^2 + 3p^4q$	<ul style="list-style-type: none"> ■ (0.31,0.31): Tipo atractivo. 	
4293902335	$q^5 + 5pq^4 + 9p^2q^3 + 9p^3q^2 + 5p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.99,0.99): Tipo atractivo. 	

Identificador	Polinomio	Puntos	Gráfica
1083671878	$4pq^4 + 5p^2q^3 + 3p^4q$	<ul style="list-style-type: none"> ■ (0,0,0) Tipo repulsivo. ■ (0.41,0.41): Tipo atractivo. 	
3688853357	$q^5 + 3pq^4 + 9p^2q^3 + 7p^3q^2 + 3p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.74,0.74): Tipo atractivo. ■ (0.99,0.99): Tipo repulsivo. 	
716079830	$3pq^4 + 4p^2q^3 + 4p^3q^2 + 4p^4q$	<ul style="list-style-type: none"> ■ (0,0,0) Tipo repulsivo. ■ (0.46,0.46): Tipo atractivo. 	
701776402	$2pq^4 + 4p^2q^3 + 2p^3q^2 + 3p^4q$	<ul style="list-style-type: none"> ■ (0,0,0): Tipo repulsivo. ■ (0.31,0.31): Tipo atractivo. 	

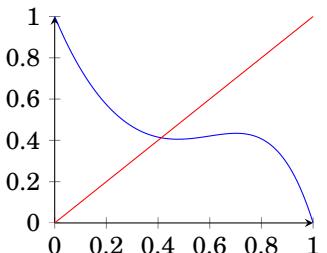
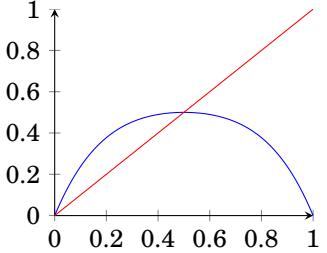
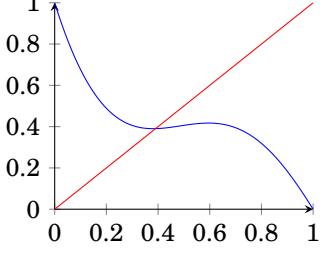
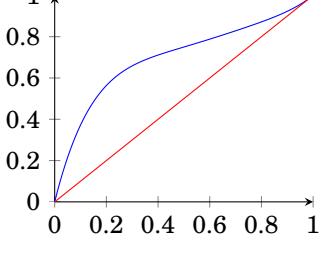
Identificador	Polinomio	Puntos	Gráfica
3430603223	$q^5 + 4pq^4 + 4p^2q^3 + 7p^3q^2 + 3p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.65, 0.65): Tipo atractivo. ■ (0.99, 0.99): Tipo repulsivo. 	
2438630838	$4pq^4 + 5p^2q^3 + 5p^3q^2 + p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.5, 0.5): Tipo atractivo. ■ (1.0, 1.0): Tipo repulsivo. 	
4054809499	$q^5 + 4pq^4 + 6p^2q^3 + 7p^3q^2 + 3p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.70, 0.70): Tipo atractivo. ■ (0.99, 0.99): Tipo repulsivo. 	
2136496127	$q^5 + 4pq^4 + 8p^2q^3 + 7p^3q^2 + 3p^4q$	<ul style="list-style-type: none"> ■ (0.62, 0.62): Tipo atractivo. 	

Identificador	Polinomio	Puntos	Gráfica
2011135287	$q^5 + 5pq^4 + 7p^2q^3 + 8p^3q^2 + 3p^4q$	<ul style="list-style-type: none"> ■ (0.64, 0.64): Tipo atractivo. 	
3892314110	$5pq^4 + 10p^2q^3 + 9p^3q^2 + 4p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.99, 0.99): Tipo atractivo. 	
3095388017	$q^5 + 3pq^4 + 8p^2q^3 + 6p^3q^2 + 3p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.69, 0.69): Tipo atractivo. ■ (0.99, 0.99): Tipo repulsivo. 	
2379996908	$2pq^4 + 9p^2q^3 + 6p^3q^2 + 3p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.69, 0.69): Tipo atractivo. ■ (0.99, 0.99): Tipo repulsivo. 	

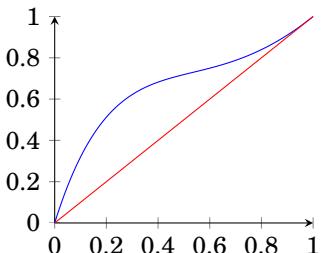
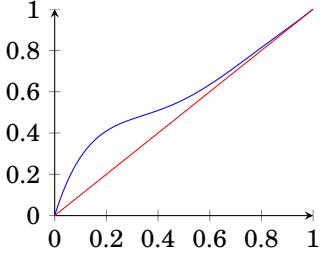
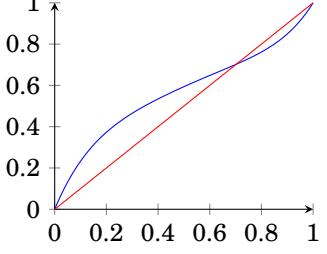
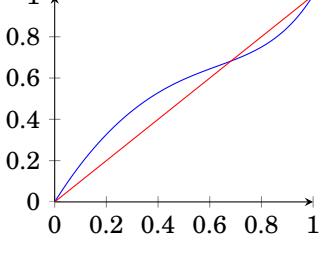
Identificador	Polinomio	Puntos	Gráfica
737934335	$q^5 + 5pq^4 + 8p^2q^3 + 8p^3q^2 + 4p^4q$	<ul style="list-style-type: none"> ■ (0.68, 0.68): Tipo atractivo. 	
902619063	$q^5 + 4pq^4 + 6p^2q^3 + 7p^3q^2 + 3p^4q$	<ul style="list-style-type: none"> ■ (0.60, 0.60): Tipo atractivo. 	
2878016454	$4pq^4 + 4p^2q^3 + 4p^3q^2 + 3p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.5, 0.5): Tipo atractivo. ■ (0.99, 0.99): Tipo repulsivo. 	
1988034548	$3pq^4 + 8p^2q^3 + 10p^3q^2 + 3p^4q$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.67, 0.67): Tipo repulsivo. 	

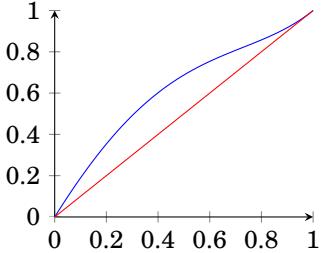
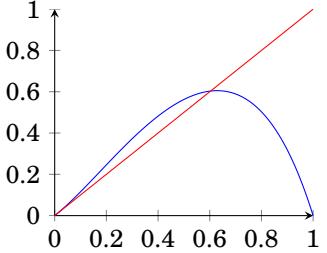
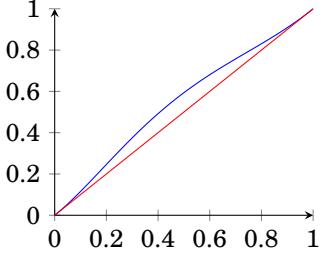
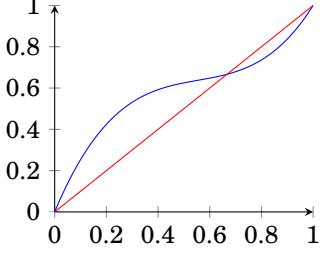
Identificador	Polinomio	Puntos	Gráfica
2547736575	$q^5 + 5pq^4 + 9p^2q^3 + 8p^3q^2 + p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.65, 0.65): Tipo atractivo. ■ (1.0, 1.0): Tipo repulsivo. 	
2729376750	$3pq^4 + 7p^2q^3 + 7p^3q^2 + 3p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.71, 0.71): Tipo atractivo. ■ (0.99, 0.99): Tipo repulsivo. 	
1946137595	$q^5 + 4pq^4 + 9p^2q^3 + 7p^3q^2 + 4p^4q$	<ul style="list-style-type: none"> ■ (0.67, 0.67): Tipo repulsivo. 	
921157631	$q^5 + 5pq^4 + 8p^2q^3 + 8p^3q^2 + 3p^4q$	<ul style="list-style-type: none"> ■ (0.65, 0.65): Tipo repulsivo. 	

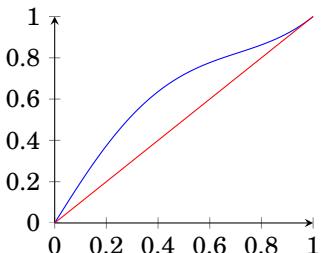
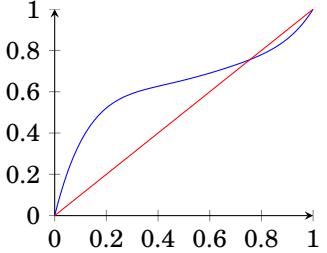
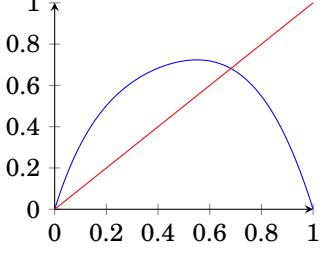
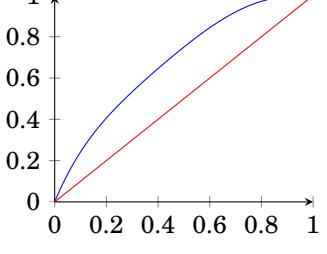
Identificador	Polinomio	Puntos	Gráfica
2248076113	$q^5 + 2pq^4 + 6p^2q^3 + 7p^3q^2 + 2p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.60, 0.60): Tipo atractivo. ■ (1.0, 1.0): Tipo repulsivo. 	
3367827048	$8p^2q^3 + 4p^3q^2 + 4p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo atractivo. ■ (0.22, 0.22): Tipo repulsivo. ■ (0.63, 0.63): Tipo atractivo. ■ (0.99, 0.99): Tipo repulsivo. 	
1323667404	$3pq^4 + 5p^2q^3 + 6p^3q^2 + 4p^4q$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.57, 0.57): Tipo atractivo. 	
3545383532	$pq^4 + 7p^2q^3 + 4p^3q^2 + p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.41, 0.41): Tipo atractivo. ■ (1.0, 1.0): Tipo repulsivo. 	

Identificador	Polinomio	Puntos	Gráfica
1486472241	$q^5 + 2pq^4 + 3p^2q^3 + 3p^3q^2 + 4p^4q$	■ (0.41, 0.41): Tipo atractivo.	
1781883580	$3pq^4 + 5p^2q^3 + 5p^3q^2 + 3p^4q$	■ (0.0, 0.0): Tipo repulsivo. ■ (0.49, 0.49): Tipo atractivo.	
514879633	$q^5 + pq^4 + 2p^2q^3 + 7p^3q^2 + 2p^4q$	■ (0.39, 0.39): Tipo atractivo.	
3756915006	$5pq^4 + 5p^2q^3 + 9p^3q^2 + 4p^4q + p^5$	■ (0.0, 0.0): Tipo repulsivo. ■ (0.99, 0.99): Tipo atractivo.	

Identificador	Polinomio	Puntos	Gráfica
4224566252	$3pq^4 + 6p^2q^3 + 6p^3q^2 + 5p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.99, 0.99): Tipo atractivo. 	
2131623934	$5pq^4 + 8p^2q^3 + 8p^3q^2 + 4p^4q$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.68, 0.68): Tipo atractivo. 	
4092976990	$5pq^4 + 8p^2q^3 + 6p^3q^2 + 4p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.99, 0.99): Tipo atractivo. 	
1934818173	$q^5 + 3pq^4 + 8p^2q^3 + 6p^3q^2 + 3p^4q$	<ul style="list-style-type: none"> ■ (0.60, 0.60): Tipo atractivo. 	

Identificador	Polinomio	Puntos	Gráfica
2868375538	$4pq^4 + 7p^2q^3 + 7p^3q^2 + 4p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.99, 0.99): Tipo atractivo. 	
2935089446	$4pq^4 + 2p^2q^3 + 7p^3q^2 + 4p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.99, 0.99): Tipo atractivo. 	
3215743430	$3pq^4 + 4p^2q^3 + 8p^3q^2 + 3p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.70, 0.70): Tipo atractivo. ■ (0.99, 0.99): Tipo repulsivo. 	
3552230320	$2pq^4 + 6p^2q^3 + 7p^3q^2 + 3p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.68, 0.68): Tipo atractivo. ■ (0.99, 0.99): Tipo repulsivo. 	

Identificador	Polinomio	Puntos	Gráfica
4121689804	$2pq^4 + 7p^2q^3 + 8p^3q^2 + 4p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.99, 0.99): Tipo atractivo. 	
800649912	$pq^4 + 6p^2q^3 + 7p^3q^2 + 4p^4q$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.60, 0.60): Tipo atractivo. 	
3740243592	$pq^4 + 6p^2q^3 + 7p^3q^2 + 4p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.99, 0.99): Tipo atractivo. 	
2902356334	$3pq^4 + 7p^2q^3 + 6p^3q^2 + 3p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.66, 0.66): Tipo atractivo. ■ (0.99, 0.99): Tipo repulsivo. 	

Identificador	Polinomio	Puntos	Gráfica
4292181480	$2pq^4 + 8p^2q^3 + 8p^3q^2 + 4p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.99, 0.99): Tipo atractivo. 	
4133616598	$5pq^4 + 3p^2q^3 + 9p^3q^2 + 3p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.75, 0.75): Tipo atractivo. ■ (0.99, 0.99): Tipo repulsivo. 	
1525415854	$4pq^4 + 6p^2q^3 + 9p^3q^2 + 4p^4q$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.68, 0.68): Tipo atractivo. 	
4276748204	$3pq^4 + 5p^2q^3 + 10p^3q^2 + 5p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.99, 0.99): Tipo crítico. 	

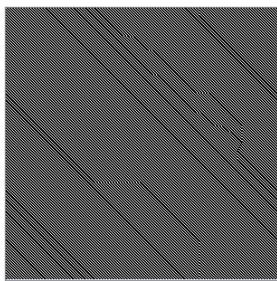
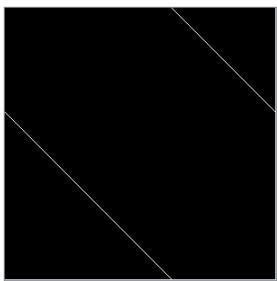
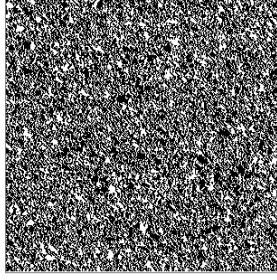
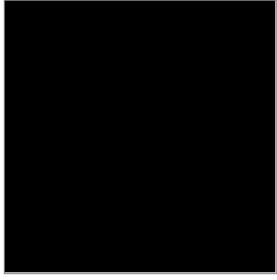
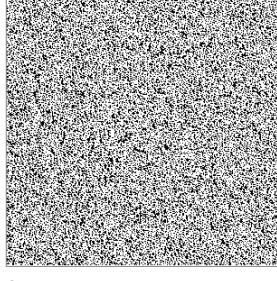
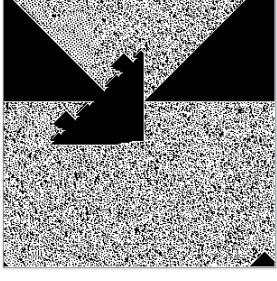
Identificador	Polinomio	Puntos	Gráfica
3975567326	$4pq^4 + 8p^2q^3 + 5p^3q^2 + 4p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.82, 0.82): Tipo atractivo. ■ (0.99, 0.99): Tipo repulsivo. 	
2901391334	$4pq^4 + 7p^2q^3 + 6p^3q^2 + 4p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.99, 0.99): Tipo atractivo. 	
3185432644	$2pq^4 + 5p^2q^3 + 6p^3q^2 + 4p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.99, 0.99): Tipo atractivo. 	
1939928251	$q^5 + 2pq^4 + 5p^2q^3 + 6p^3q^2 + 4p^4q$	<ul style="list-style-type: none"> ■ (0.56, 0.56): Tipo atractivo. 	

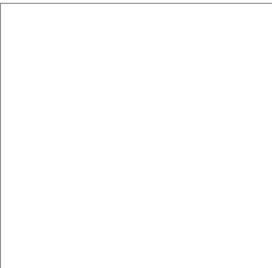
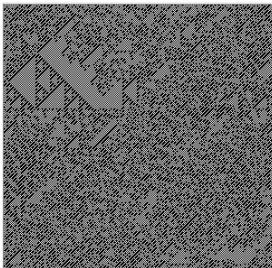
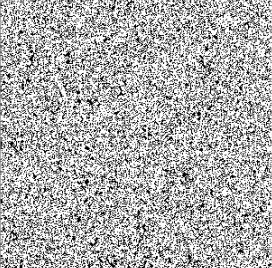
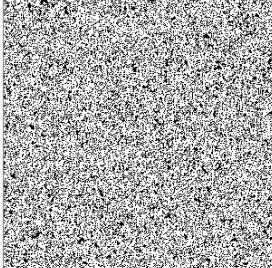
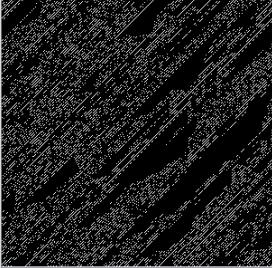
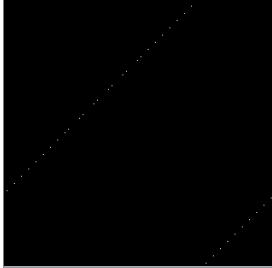
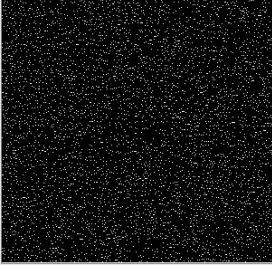
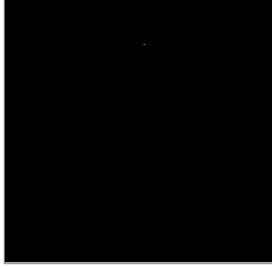
Identificador	Polinomio	Puntos	Gráfica
4103131697	$q^5 + pq^4 + 4p^2q^3 + 3p^3q^2 + 4p^4q + p^5$	<ul style="list-style-type: none"> ■ (0.40, 0.40): Tipo atractivo. ■ (0.99, 0.99): Tipo repulsivo. 	<p>A graph on a coordinate plane where both axes range from 0 to 1. A blue curve starts at (0,1), dips below the line y=x, crosses it at approximately (0.4, 0.4), dips again, crosses the line again at approximately (0.99, 0.99), and then rises towards (1,1). A straight red line represents the identity function y=x.</p>
1966165975	$q^5 + 5pq^4 + 5p^2q^3 + 6p^3q^2 + 2p^4q$	<ul style="list-style-type: none"> ■ (0.55, 0.55): Tipo atractivo. 	<p>A graph on a coordinate plane where both axes range from 0 to 1. A blue curve starts at (0,1), dips below the line y=x, crosses it at approximately (0.55, 0.55), and then rises towards (1,1). A straight red line represents the identity function y=x.</p>
1688232692	$2pq^4 + 3p^2q^3 + 5p^3q^2 + 3p^4q$	<ul style="list-style-type: none"> ■ (0.0, 0.0): Tipo repulsivo. ■ (0.34, 0.34): Tipo atractivo. 	<p>A graph on a coordinate plane where both axes range from 0 to 1. A blue curve starts at (0,1), dips below the line y=x, crosses it at approximately (0.34, 0.34), and then rises towards (1,1). A straight red line represents the identity function y=x.</p>

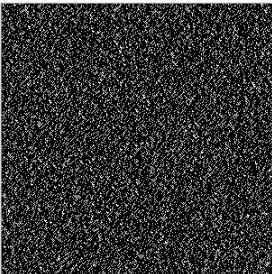
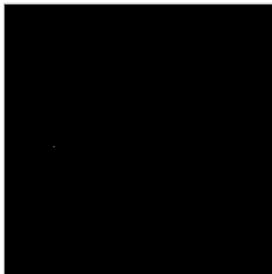
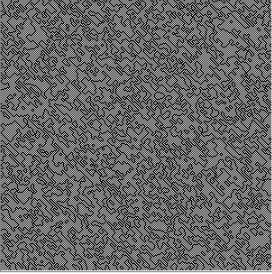
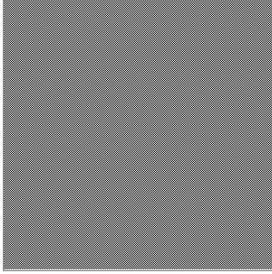
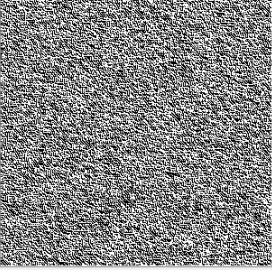
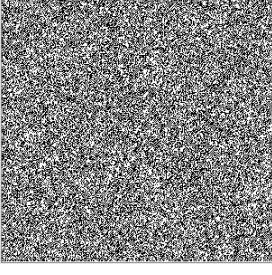
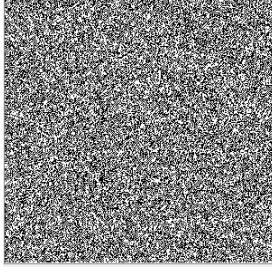
Tabla 9.1: Asociación de reglas y polinomios

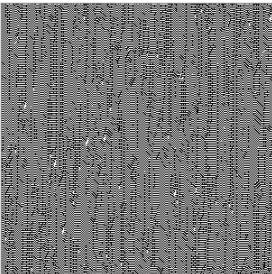
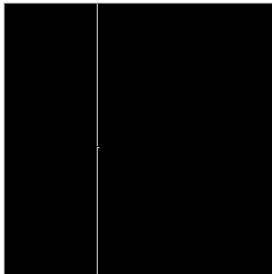
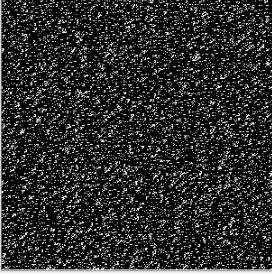
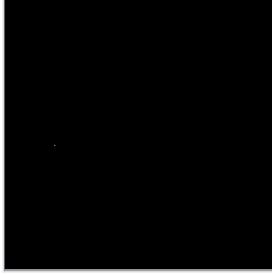
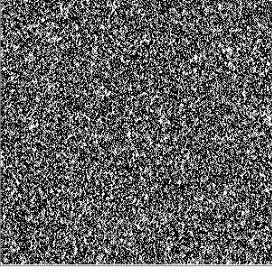
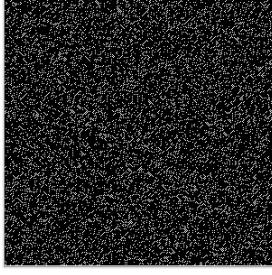
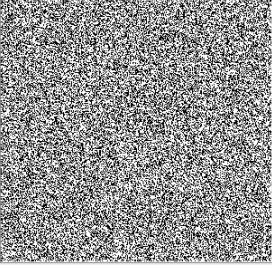
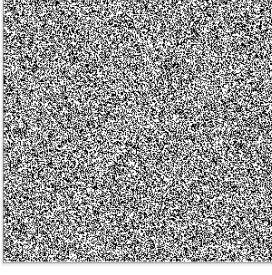
9.2. Asociación de reglas y densidades

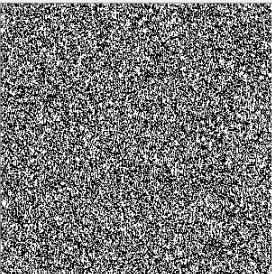
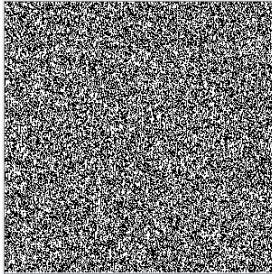
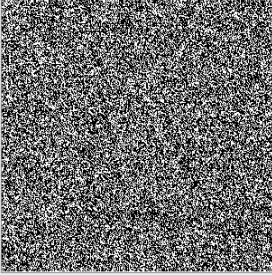
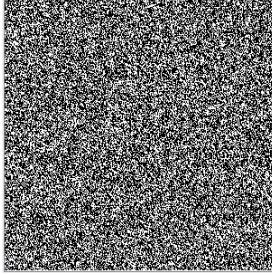
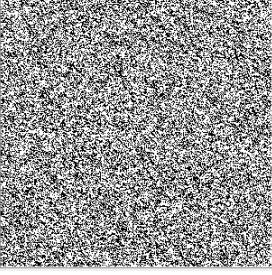
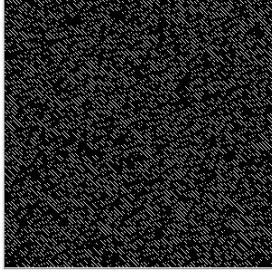
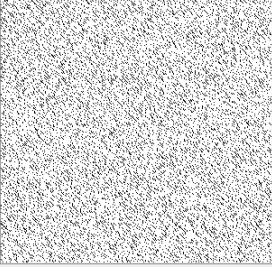
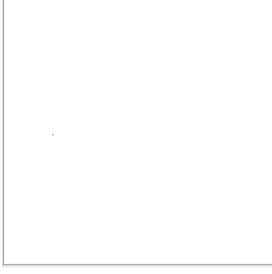
Ahora, para las mismas reglas exploradas anteriormente, se ilustra su comportamiento iniciando el simulador con un espacio de evolución de 300×300 células con diferentes densidades (al 50% y con una sola célula viva al centro). Dependiendo de la regla y densidad de células vivas se pueden encontrar diferentes patrones y comportamientos.

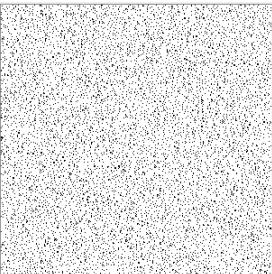
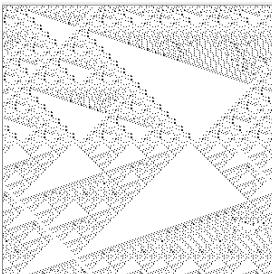
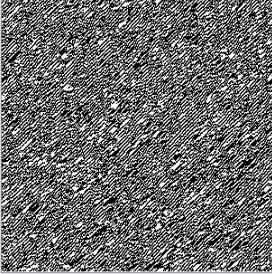
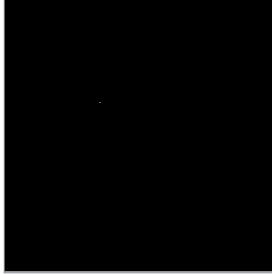
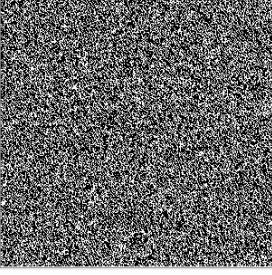
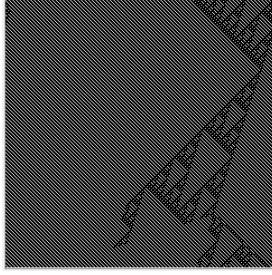
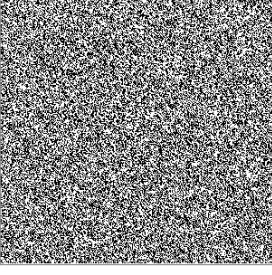
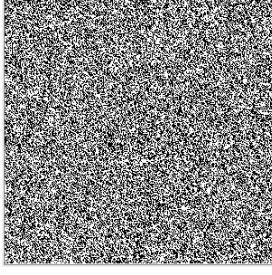
Identificador	50 % de células vivas	1 célula viva
539312132	 Generación número: 500	 Generación número: 500
2436109	 Generación número: 500	 Generación número: 500
1853030134	 Generación número: 500	 Generación número: 500

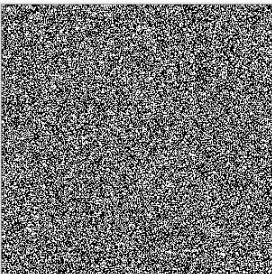
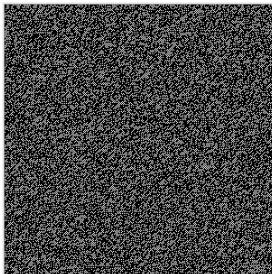
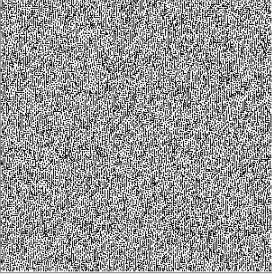
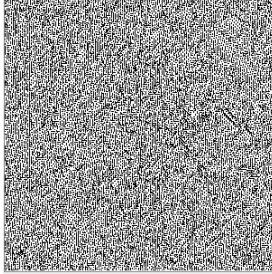
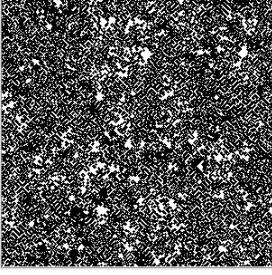
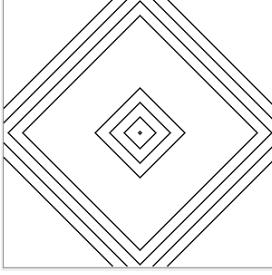
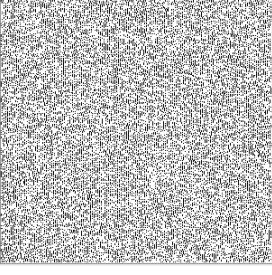
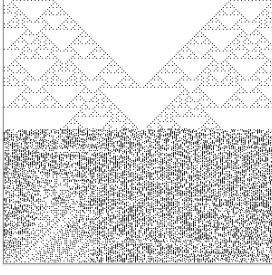
Identificador	50 % de células vivas	1 célula viva
4189052334	 Generación número: 20	 Generación número: 500
2111306941	 Generación número: 500	 Generación número: 500
1077936134	 Generación número: 500	 Generación número: 500
8454144	 Generación número: 500	 Generación número: 100

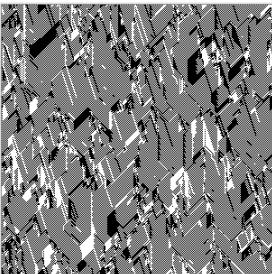
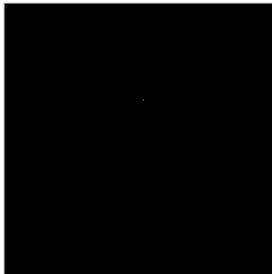
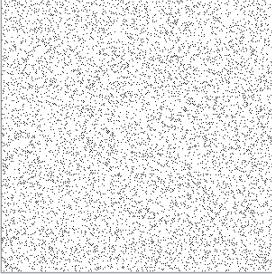
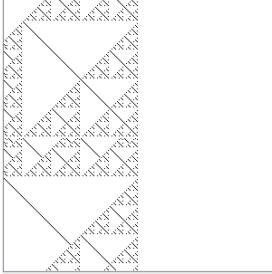
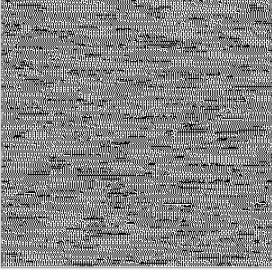
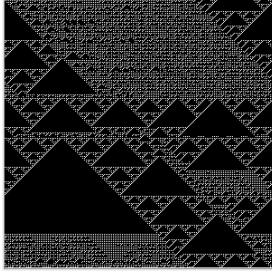
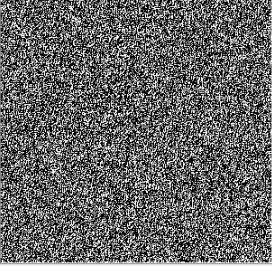
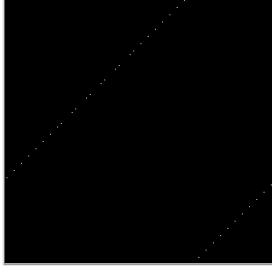
Identificador	50% de células vivas	1 célula viva
50627360	 Generación número: 500	 Generación número: 100
1884435473	 Generación número: 100	 Generación número: 400
3943755596	 Generación número: 100	 Generación número: 250
2422207638	 Generación número: 100	 Generación número: 500

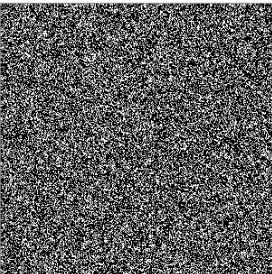
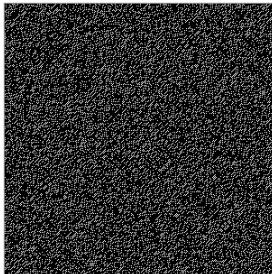
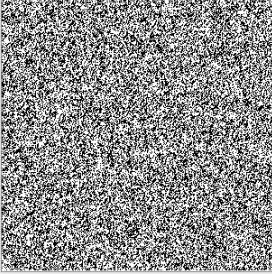
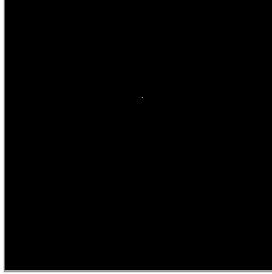
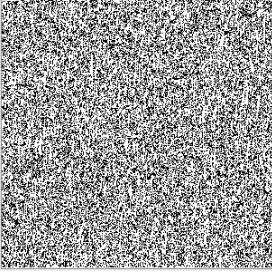
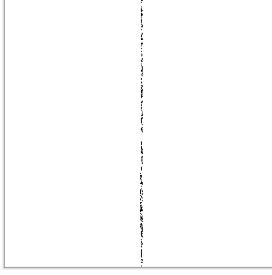
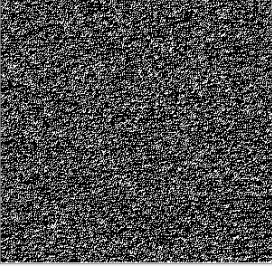
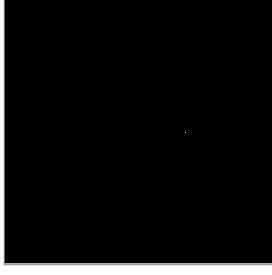
Identificador	50 % de células vivas	1 célula viva
177146253	 Generación número: 300	 Generación número: 700
2585170240	 Generación número: 150	 Generación número: 100
1686636573	 Generación número: 100	 Generación número: 500
1540729301	 Generación número: 100	 Generación número: 750

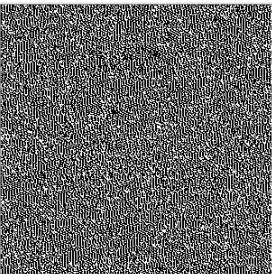
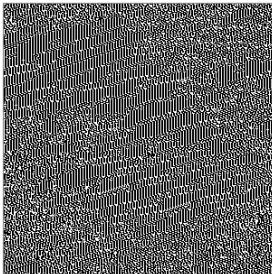
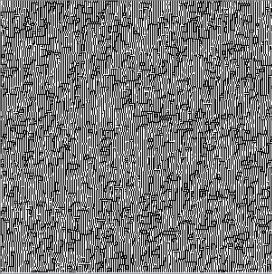
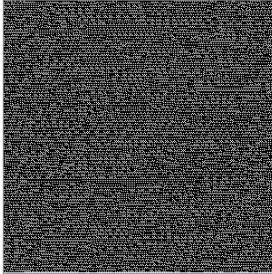
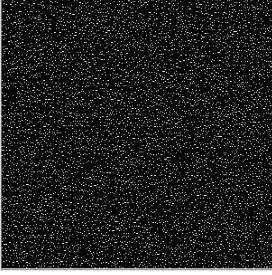
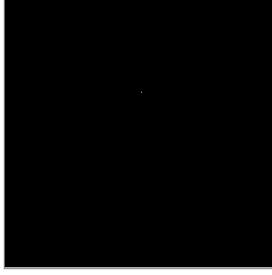
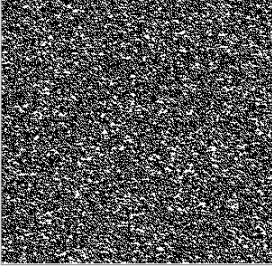
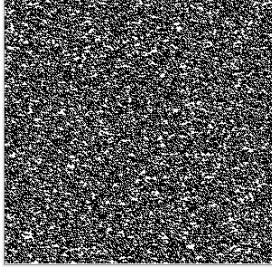
Identificador	50 % de células vivas	1 célula viva
2737841485	 Generación número: 100	 Generación número: 500
457382132	 Generación número: 100	 Generación número: 600
1457500966	 Generación número: 100	 Generación número: 500
3996909215	 Generación número: 100	 Generación número: 100

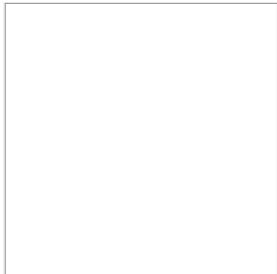
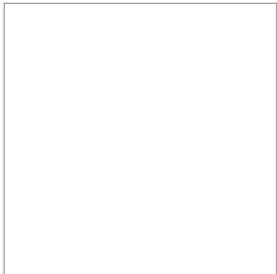
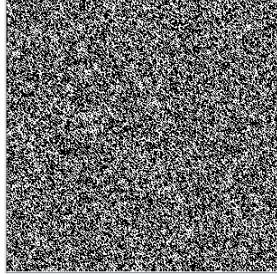
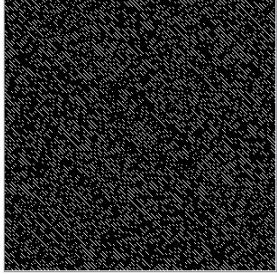
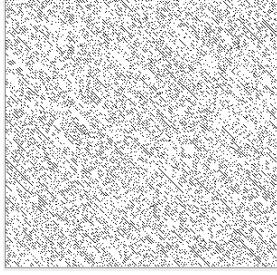
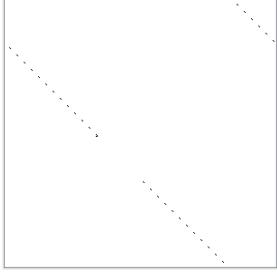
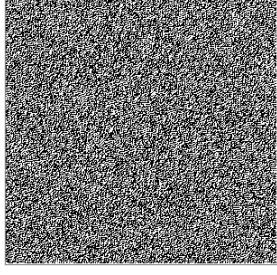
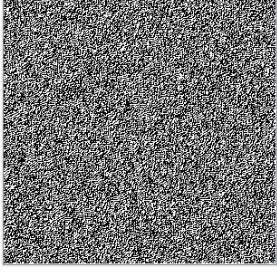
Identificador	50 % de células vivas	1 célula viva
4285263798	 Generación número: 100	 Generación número: 500
4223663	 Generación número: 100	 Generación número: 100
3085644	 Generación número: 100	 Generación número: 600
2049698883	 Generación número: 100	 Generación número: 350

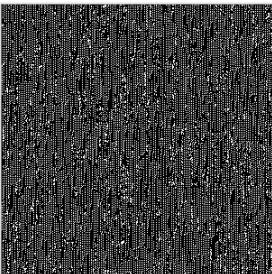
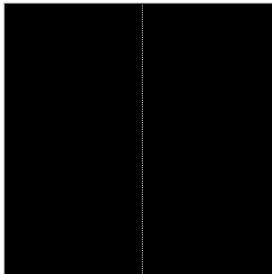
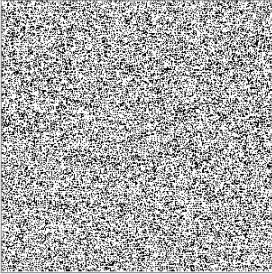
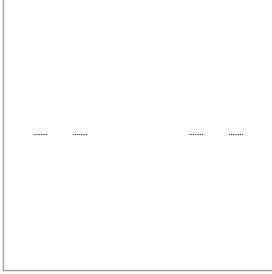
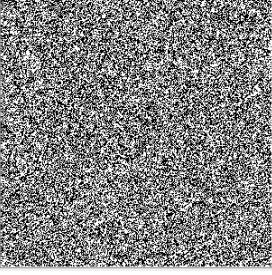
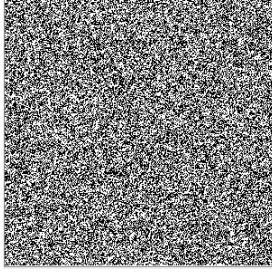
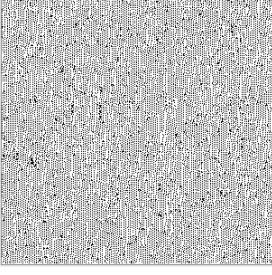
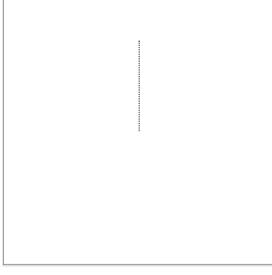
Identificador	50 % de células vivas	1 célula viva
341699601	 Generación número: 100	 Generación número: 400
871013027	 Generación número: 100	 Generación número: 500
2147483649	 Generación número: 100	 Generación número: 600
3480215414	 Generación número: 100	 Generación número: 300

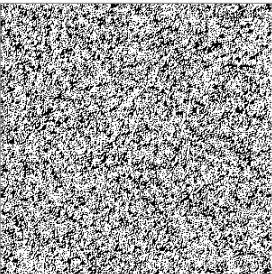
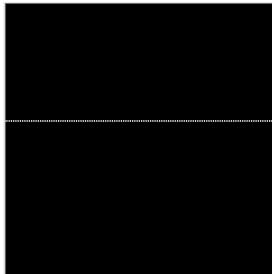
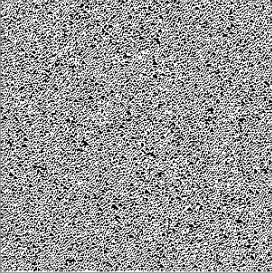
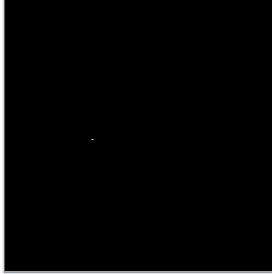
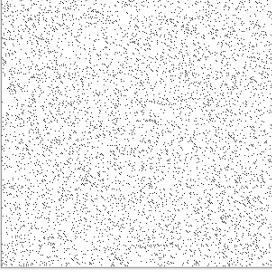
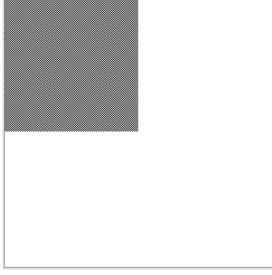
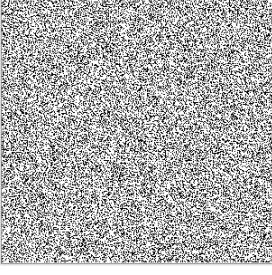
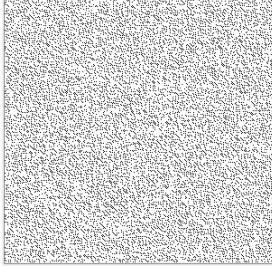
Identificador	50 % de células vivas	1 célula viva
2422207	 Generación número: 100	 Generación número: 50
4143971324	 Generación número: 50	 Generación número: 500
1867275334	 Generación número: 100	 Generación número: 600
309203110	 Generación número: 100	 Generación número: 500

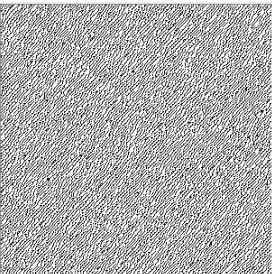
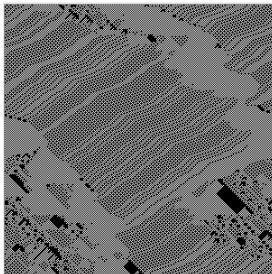
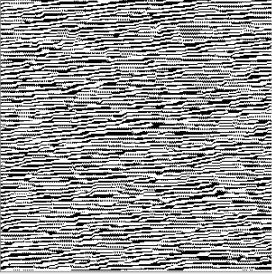
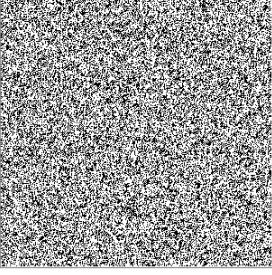
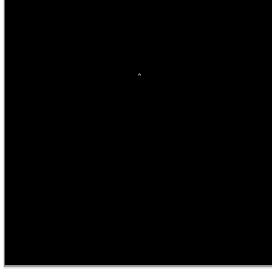
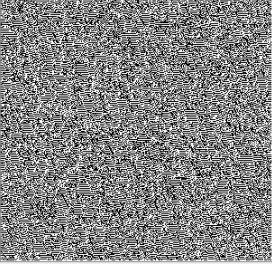
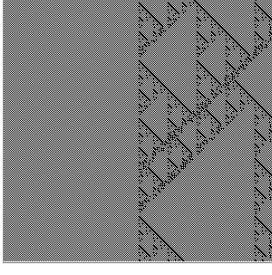
Identificador	50 % de células vivas	1 célula viva
2702504454	 Generación número: 100	 Generación número: 650
1152074856	 Generación número: 100	 Generación número: 50
3047921227	 Generación número: 100	 Generación número: 400
687883884	 Generación número: 100	 Generación número: 50

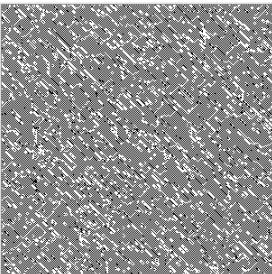
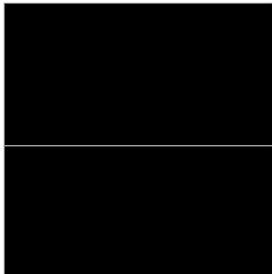
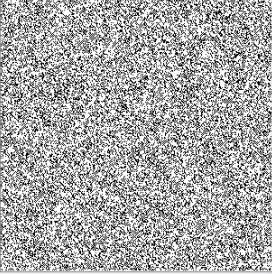
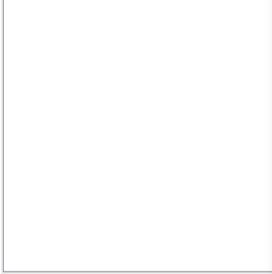
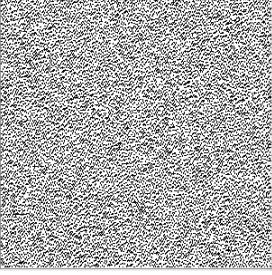
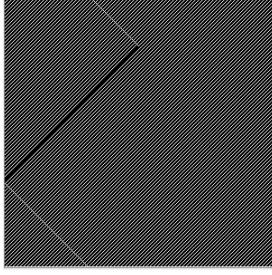
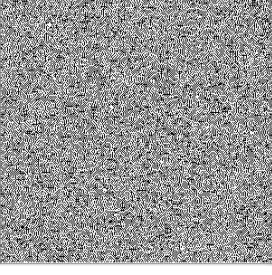
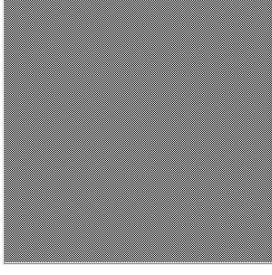
Identificador	50 % de células vivas	1 célula viva
1103116065	 Generación número: 50	 Generación número: 500
33938756	 Generación número: 50	 Generación número: 500
2149679104	 Generación número: 50	 Generación número: 50
1745944769	 Generación número: 50	 Generación número: 550

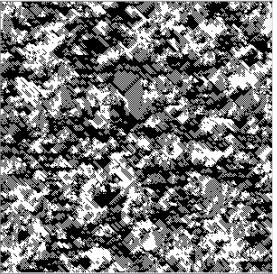
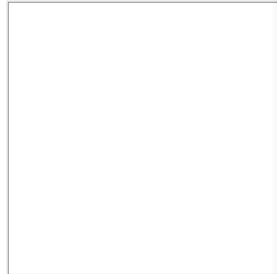
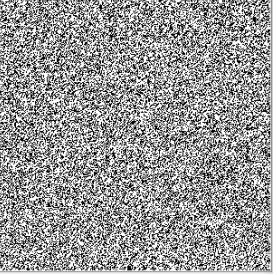
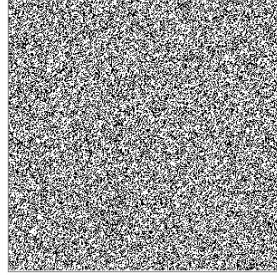
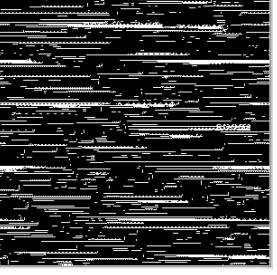
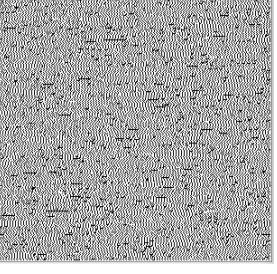
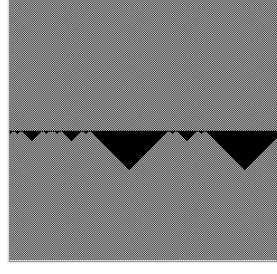
Identificador	50 % de células vivas	1 célula viva
4293902335	 Generación número: 5	 Generación número: 1
1083671878	 Generación número: 50	 Generación número: 350
3688853357	 Generación número: 50	 Generación número: 250
716079830	 Generación número: 50	 Generación número: 1100

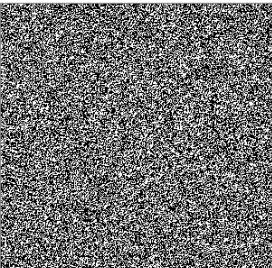
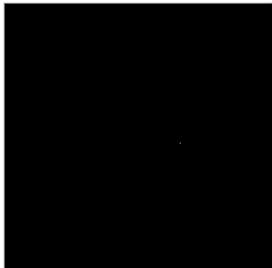
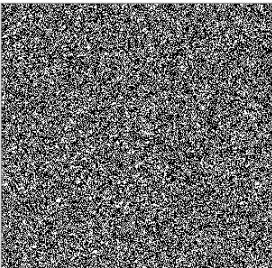
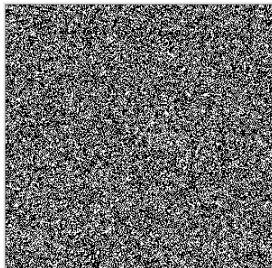
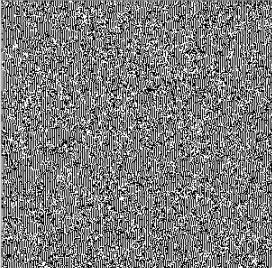
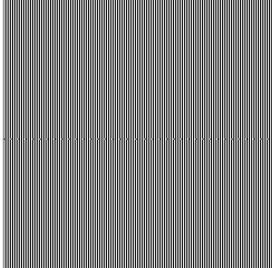
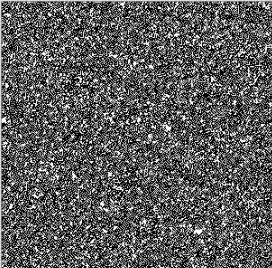
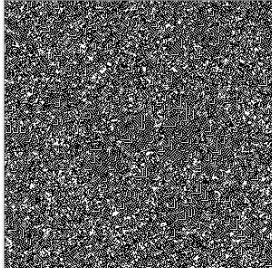
Identificador	50 % de células vivas	1 célula viva
701776402	 Generación número: 100	 Generación número: 350
3430603223	 Generación número: 50	 Generación número: 200
2438630838	 Generación número: 50	 Generación número: 450
4054809499	 Generación número: 100	 Generación número: 100

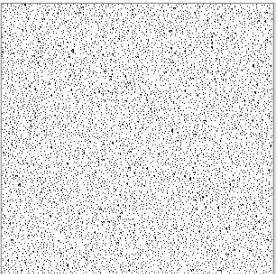
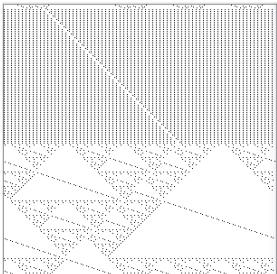
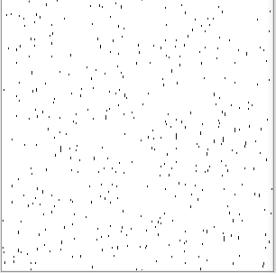
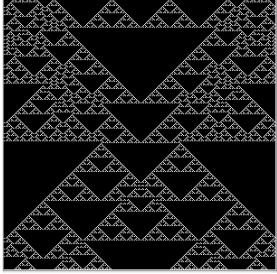
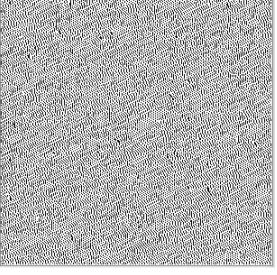
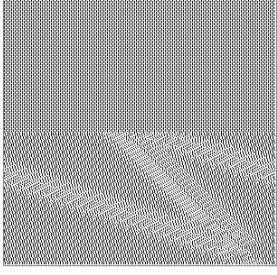
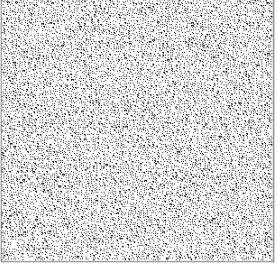
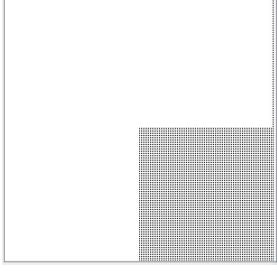
Identificador	50 % de células vivas	1 célula viva
2136496127	 Generación número: 50	 Generación número: 650
2011135287	 Generación número: 50	 Generación número: 50
3892314110	 Generación número: 50	 Generación número: 350
3095388017	 Generación número: 50	 Generación número: 650

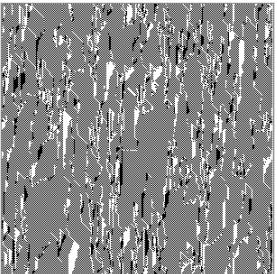
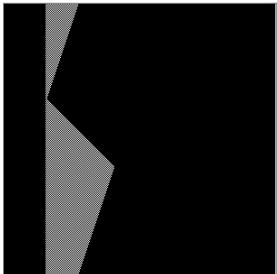
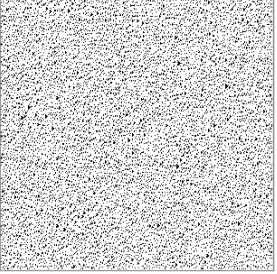
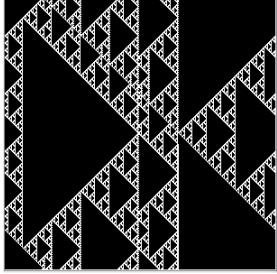
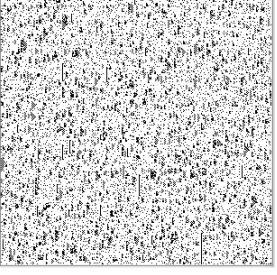
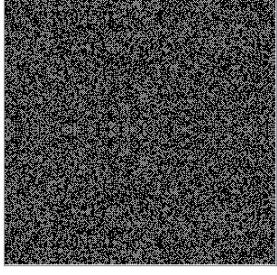
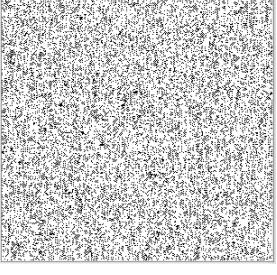
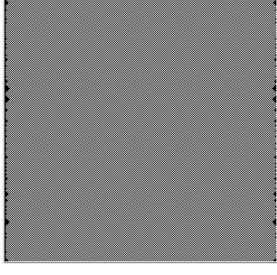
Identificador	50 % de células vivas	1 célula viva
2379996908	 Generación número: 50	 Generación número: 800
737934335	 Generación número: 50	 Generación número: 1
902619063	 Generación número: 50	 Generación número: 70
2878016454	 Generación número: 50	 Generación número: 350

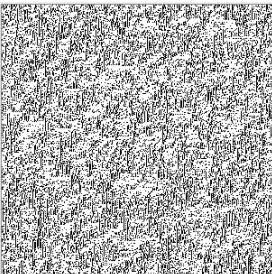
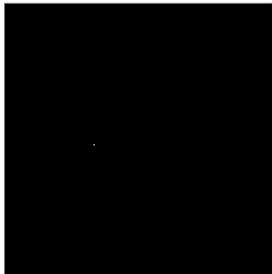
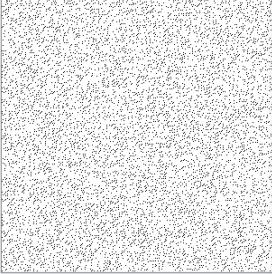
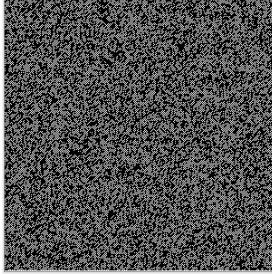
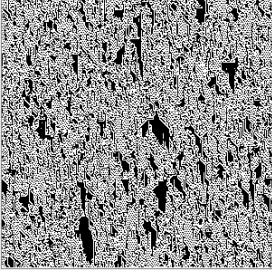
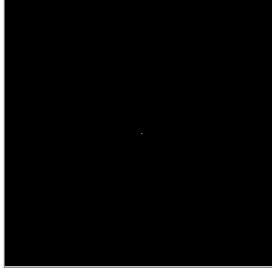
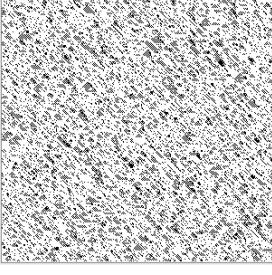
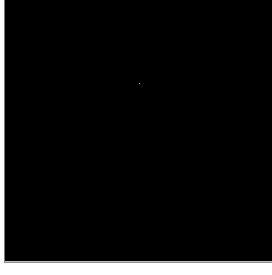
Identificador	50 % de células vivas	1 célula viva
1988034548	 Generación número: 100	 Generación número: 200
2547736575	 Generación número: 50	 Generación número: 1
2729376750	 Generación número: 50	 Generación número: 500
1946137595	 Generación número: 50	 Generación número: 650

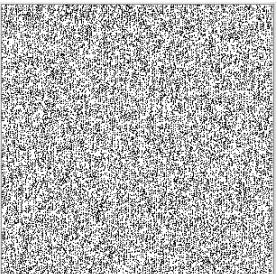
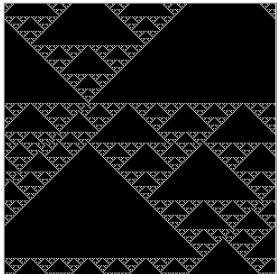
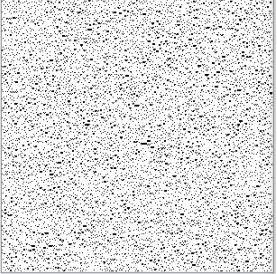
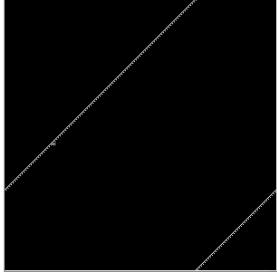
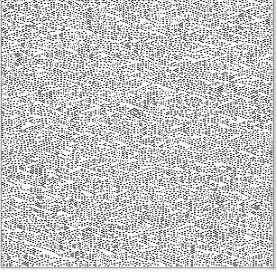
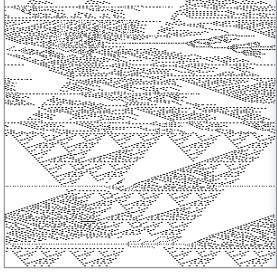
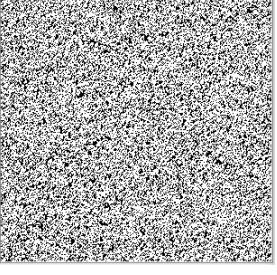
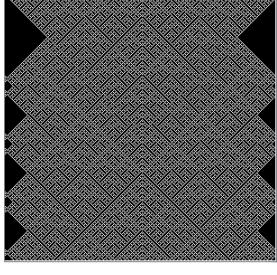
Identificador	50 % de células vivas	1 célula viva
921157631	 Generación número: 100	 Generación número: 1
2248076113	 Generación número: 50	 Generación número: 1000
3367827048	 Generación número: 400	 Generación número: 1
1323667404	 Generación número: 50	 Generación número: 500

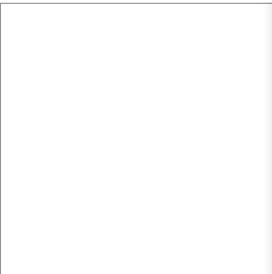
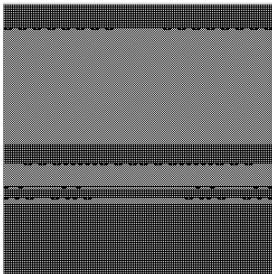
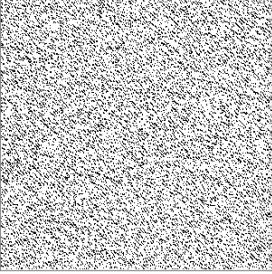
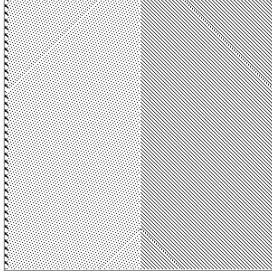
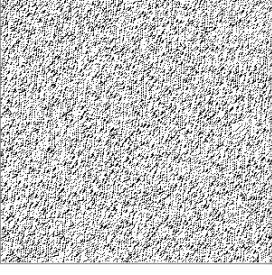
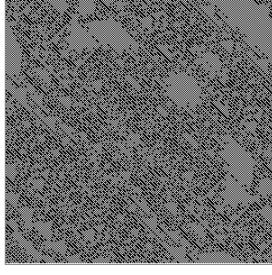
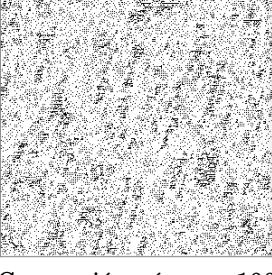
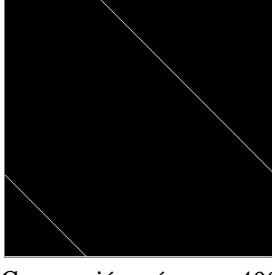
Identificador	50 % de células vivas	1 célula viva
3545383532	 Generación número: 50	 Generación número: 50
1486472241	 Generación número: 50	 Generación número: 300
1781883580	 Generación número: 50	 Generación número: 700
514879633	 Generación número: 50	 Generación número: 400

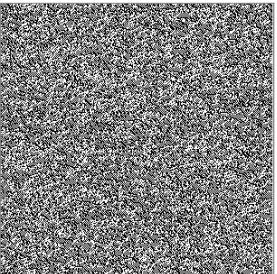
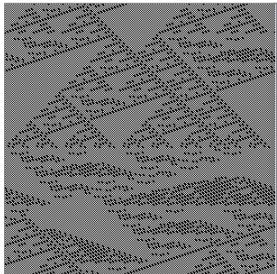
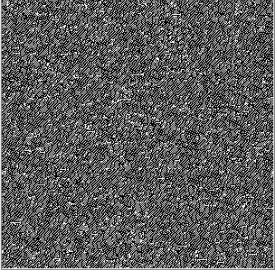
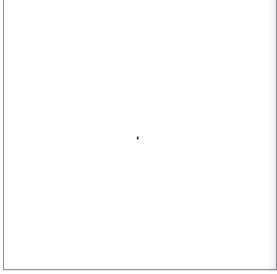
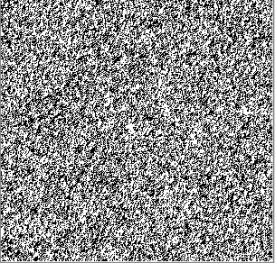
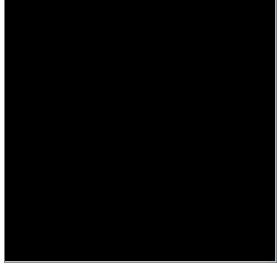
Identificador	50 % de células vivas	1 célula viva
3756915006	 Generación número: 50	 Generación número: 350
4224566252	 Generación número: 50	 Generación número: 550
2131623934	 Generación número: 50	 Generación número: 1
4092976990	 Generación número: 50	 Generación número: 350

Identificador	50 % de células vivas	1 célula viva
1934818173	 Generación número: 150	 Generación número: 550
2868375538	 Generación número: 50	 Generación número: 550
2935089446	 Generación número: 50	 Generación número: 300
3215743430	 Generación número: 50	 Generación número: 650

Identificador	50 % de células vivas	1 célula viva
3552230320	 Generación número: 50	 Generación número: 50
4121689804	 Generación número: 50	 Generación número: 550
800649912	 Generación número: 100	 Generación número: 50
3740243592	 Generación número: 50	 Generación número: 50

Identificador	50 % de células vivas	1 célula viva
2902356334	 Generación número: 50	 Generación número: 500
4292181480	 Generación número: 50	 Generación número: 400
4133616598	 Generación número: 50	 Generación número: 400
1525415854	 Generación número: 50	 Generación número: 300

Identificador	50 % de células vivas	1 célula viva
4276748204	 Generación número: 10	 Generación número: 700
3975567326	 Generación número: 50	 Generación número: 500
2901391334	 Generación número: 50	 Generación número: 350
3185432644	 Generación número: 100	 Generación número: 400

Identificador	50 % de células vivas	1 célula viva
1939928251	 Generación número: 100	 Generación número: 400
4108131697	 Generación número: 100	 Generación número: 50
1966165975	 Generación número: 100	 Generación número: 10

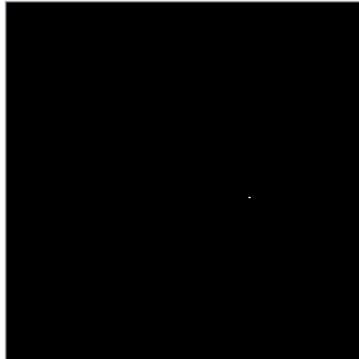
Identificador	50 % de células vivas	1 célula viva
1688232692	 Generación número: 100	 Generación número: 50

Tabla 9.2: Asociación de reglas y densidades

10

Clasificación

10.1. Clasificación de Wolfram

Basándonos en el comportamiento y polinomio característico de cada regla de evolución se ha realizado una clasificación para las reglas exploradas.

Clase I	Clase II	Clase III	Clase IV
4293902335	539312132	2147483649	3943755596
	2436109	1946137595	3480215414
	1853030134	921157631	1867275334
	4189052334		309203110
	2111306941		2702504454
	1077936134		1152074856
	8454144		3047921227
	50627360		3688853357
	1884435473		701776402
	2422207638		3430603223
	177146253		4054809499
	2585170240		3095388017
	1686636573		2379996908
	1540729301		2878016454
	2737841485		3367827048
	457382132		3552230320
	1457500966		2902356334
	3996909215		4133616598
	4285263798		3185432644
	4223663		4103131697
	3085644		687883884
	2049698883		3215743430
	341699601		800649912
	871013027		1688232692
	2422207		
	4143971324		
	1103116065		
	33938756		
	2149679104		
	1745944769		
	1083671878		
	716079830		
	2438630838		
	2136496127		
	2011135287		
	3892314110		
	737934335		
	902619063		
	2729376750		
	2248076113		
	1323667404		
	3545383532		
	1486472241		
	1781883580		
	514879633		

Clase I	Clase II	Clase III	Clase IV
	3756915006		
	4224566252		
	2131623934		
	4092976990		
	1934818173		
	2868375538		
	4121689804		
	3740243592		
	4292181480		
	1525415854		
	4276748204		
	3975567326		
	2901391334		
	1939928251		
	1966165975		
	2547736575		
	2935089446		
	1988034548		

Tabla 10.1: Clasificación de Wolfram

11

Casos de estudio

11.1. Analizando la regla 1688232692

Esta regla posee comportamientos diferentes dependiendo de la densidad de la configuración inicial, por ejemplo si esta es menor o igual a 1% parece que solo existen células que se desplazan de izquierda a derecha pero no hay interacción alguna, es decir, todas las células se comportan como *gliders* este ejemplo se muestra en Fig. 11.1.1.

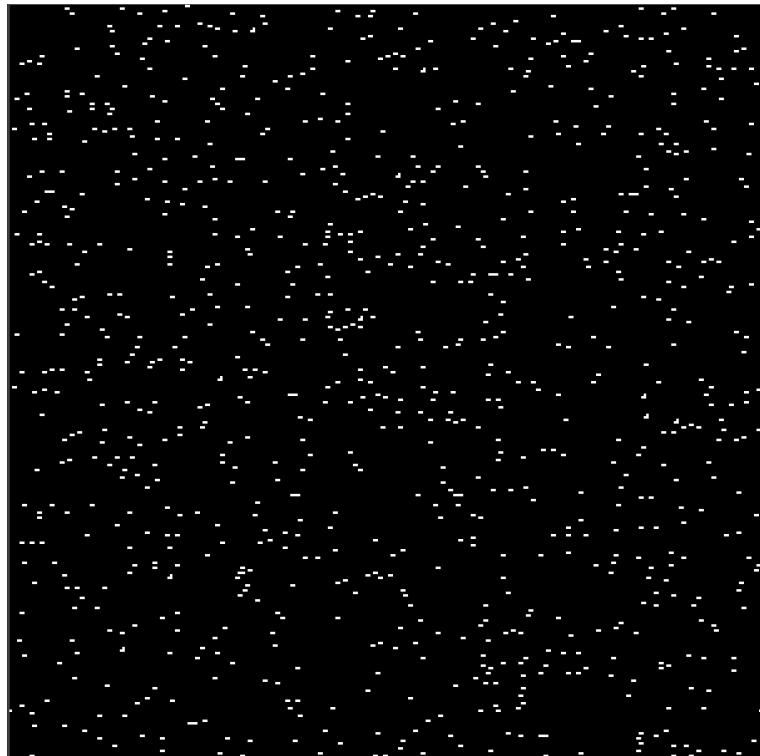


Fig. 11.1.1: Simulación de la regla 1688232692 con densidad inicial de 0.9%
Fuente: Realización propia.

Por otra parte, si incrementamos la densidad inicial de esta regla tenemos comportamientos más intere-

santes si la densidad es mayor a 1% algunas células se desplazan hasta que colisionan con otras formando ramificaciones que se mantienen creciendo hasta que no hay más colisiones entre las células, además existe cierto periodo en el que las células se contraen y se desplazan hacia la derecha, este periodo cambia dependiendo de la configuración inicial, a menor densidad de células en la configuración inicial mayor es el periodo mientras que si la densidad de la configuración inicial es grande este periodo s muy pequeño. Lo anterior podemos observarlo en Fig. 11.1.2 donde las células cuyo color no es negro ni blanco representan células que se han mantenido vivas durante varias generaciones.

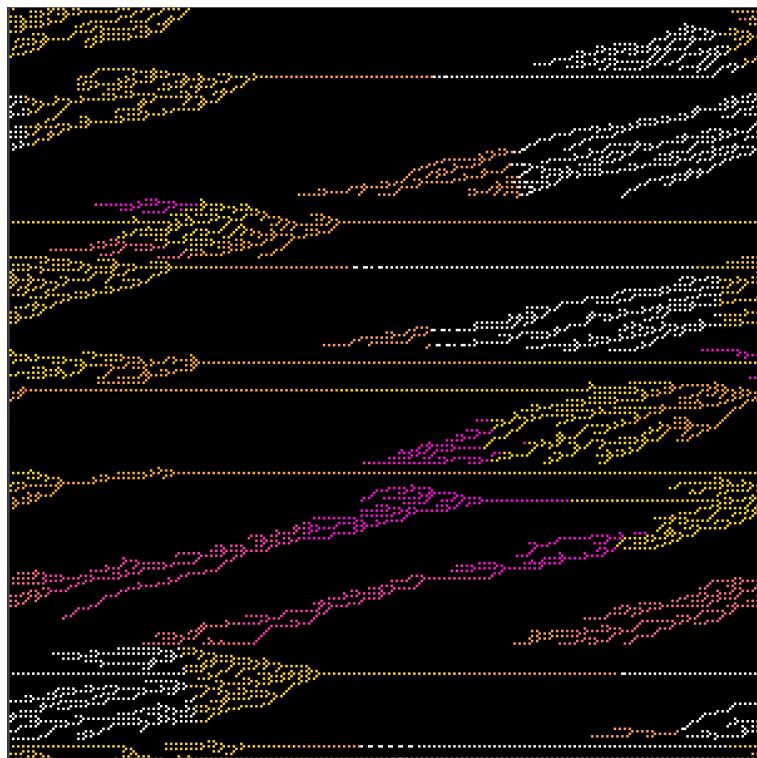


Fig. 11.1.2: Simulación de la regla 1688232692 con densidad inicial de 10%

Fuente: Realización propia utilizando el simulador vNCASimulator .

Esta regla podría tener relación alguna con un tema de botánica, en concreto la corteza. La corteza se entiende como la capa que cubre y protege la madera de los tallos de árboles y los patrones que están presentes en la misma se asemejan a esta regla.

En Fig. 11.1.3 se observa una comparación entre la regla analizada y la corteza de pino, las células en color amarillo o bien en rojo son células que han estado vivas durante varias generaciones.

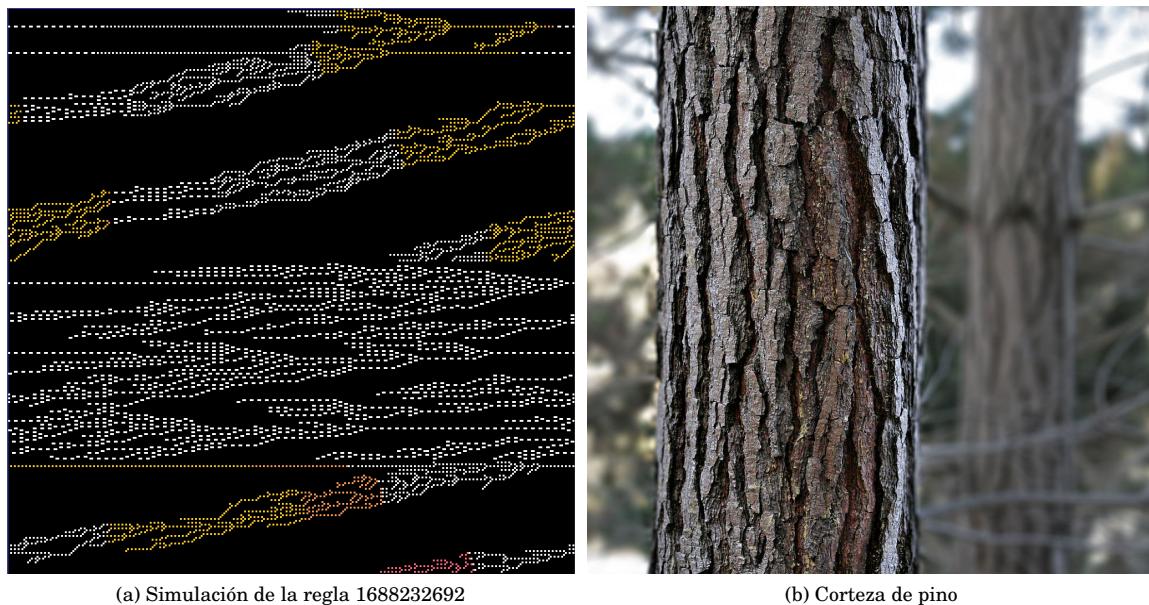


Fig. 11.1.3: Comparación de la regla y corteza de pino
 Fuente: Realización propia. Bark of Radiata Pine tree by fir0002. License: CC BY-NC 3.0

El polinomio característico de esta regla es $2p(1-p)^4 + 3p^2(1-p)^3 + 5p^3(1-p)^2 + 3p^4(1-p)$ y como podemos observar en Fig. 11.1.4 existen dos puntos de intersección entre la recta identidad y el polinomio, teniendo primeramente un punto fijo del tipo *repulsivo* cerca del origen, es decir, cerca de $(0.0, 0.0)$ mientras que existe otro punto fijo en $(0.34, 0.34)$ pero este es del tipo *atractivo*. El primer punto indica que con densidades muy pequeñas lo que ocurrirá en la generación $t+1$ es impredecible, podría mantenerse la misma densidad de células vivas o bien las células podrían pasar a un estado 0 (muerto) o el último caso es que nazcan nuevas células y como se mencionó anteriormente esto es apreciable con densidades más pequeñas a 1%. Si la densidad es mayor a 1% hay cierta estabilidad en la n -ésima generación, la densidad se aproxima a 0.70 tal y como se puede observar en Fig. 11.1.5 pero la entropía tiende al valor del segundo punto fijo, es decir, $(0.34, 0.34)$ tal y como se observa en Fig. 11.1.6. Esta regla es clasificada como compleja primeramente porque como se ha mencionado tenemos dos tipos de puntos fijos, tanto del tipo *atractivo* como del tipo *repulsivo* y además porque siempre existe interacción entre las células y además se forman ciertos patrones pero sin tener un orden o bien periodo definido.

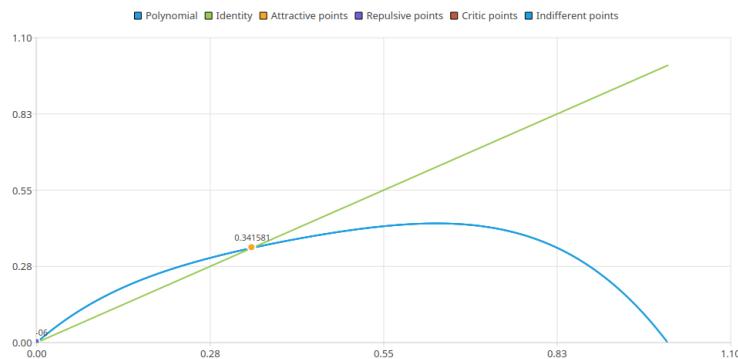


Fig. 11.1.4: Polinomio característico de la regla 1688232692

Fuente: Realización propia.

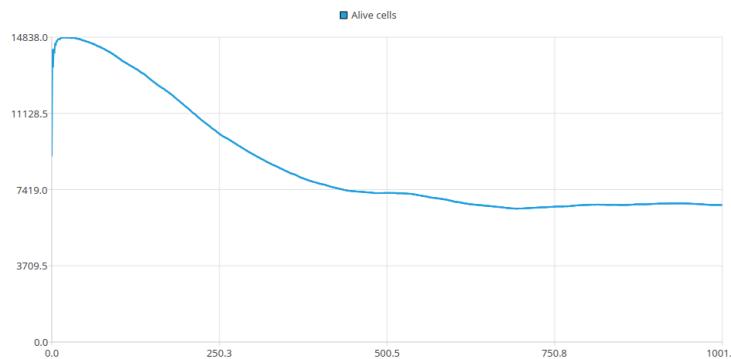


Fig. 11.1.5: Población de la regla 1688232692 hasta la generación 1000

Fuente: Realización propia.

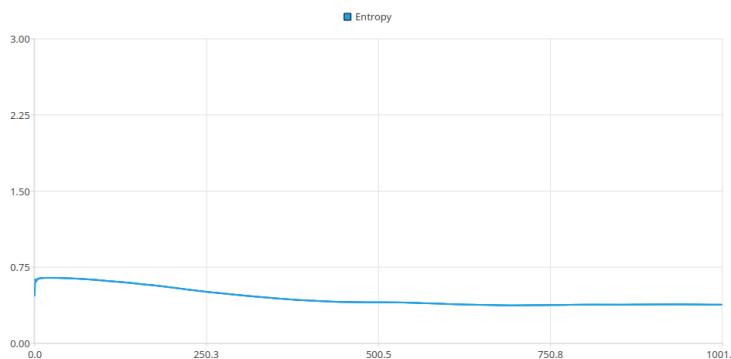


Fig. 11.1.6: Entropía de la regla 1688232692 hasta la generación 1000

Fuente: Realización propia.

11.2. Analizando la regla 2878016454

Esta regla ha sido clasificada como del tipo complejo, su polinomio característico $4pq^4 + 4p^2q^3 + 4p^3q^2 + 3p^4q + p^5$ y la recta identidad intersectan en tres puntos. Esta regla tiene diferentes comportamientos dependiendo de la configuración inicial, incluso si solo existe una célula en el espacio de evolución el comportamiento es diferente a si se tuviera una densidad inicial mayor. Sin importar la densidad inicial del AC se generan movimientos tanto horizontales, verticales y en diagonal existe interacción entre las células, a excepción del caso particular cuando hay una sola célula en el espacio de evolución. En la siguiente figura podemos observar el polinomio característico, la recta identidad y los puntos de intersección entre ambas funciones.

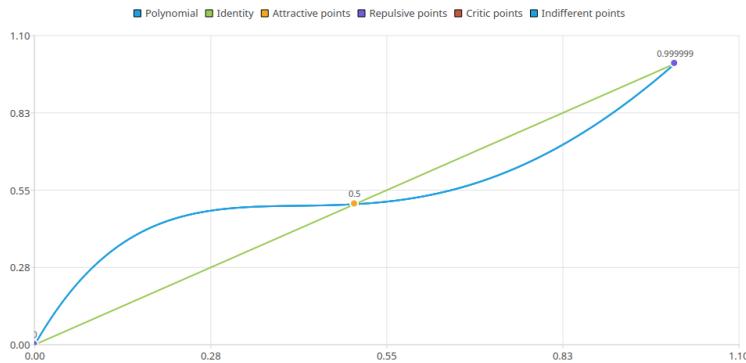


Fig. 11.2.7: Polinomio característico de la regla 2878016454

Fuente: Realización propia.

Esta regla tiene tres puntos fijos tal y como se observa en Fig. 11.2.7, el primero en el origen, es decir, en el par ordenado $(0.0, 0.0)$ este punto es del tipo repulsivo además de existir otro punto fijo de la misma clase pero en el par ordenado $(1.0, 1.0)$, el que dichos puntos sean del tipo repulsivo significa que es impredecible saber que ocurrirá con esta densidad o bien con valores cercanos a esta, tanto podrían nacer células como podrían morir o bien es posible que solo exista caos. Hay un tercer punto fijo en el par ordenado $(0.5, 0.5)$ pero a diferencia de los previamente mencionados este es del tipo atractivo, es decir que teóricamente el sistema debe estabilizarse en dicho punto en la n -ésima generación.

Como se mencionó previamente dependiendo de la configuración inicial se obtienen comportamientos diferentes, por ejemplo si la configuración inicial tiene una densidad menor al 0.01% o si dicha densidad es mayor a 99% en el espacio de evolución se generan patrones triangulares los cuales se mantienen en expansión hasta que el espacio de evolución está lleno por dichos patrones y se alcanza estabilidad, podemos observar dicho comportamiento en Fig. 11.2.8.

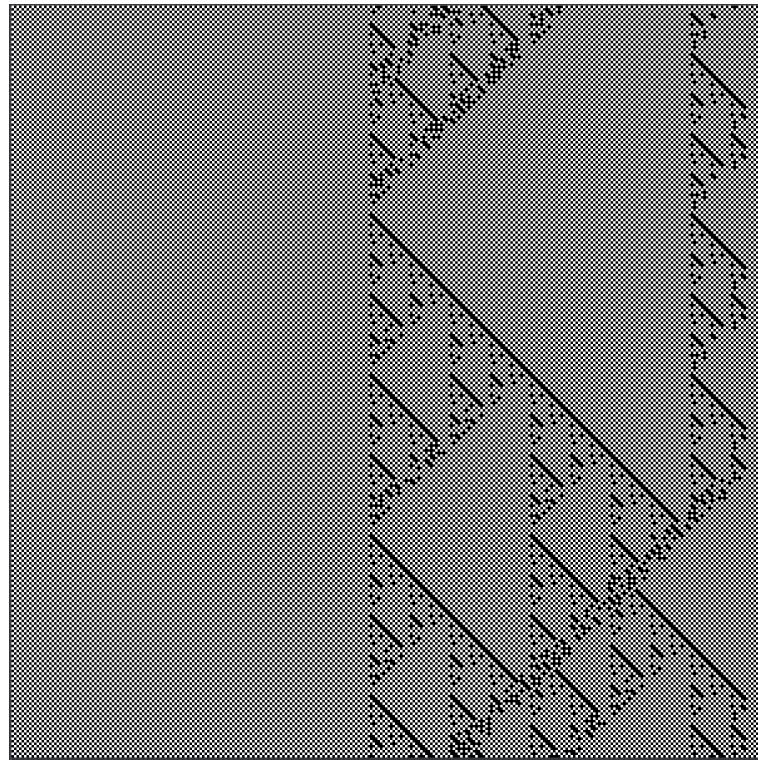


Fig. 11.2.8: Simulación de la regla 2878016454 con una sola célula viva tras 500 generaciones
Fuente: Realización propia.

En Fig. 11.2.9 podemos observar el comportamiento de la densidad de población y la entropía, como podemos apreciar la densidad de población se aproxima al punto fijo atractor que se encuentra en $(0.5, 0.5)$ mientras que la entropía tiende a el valor de 1.

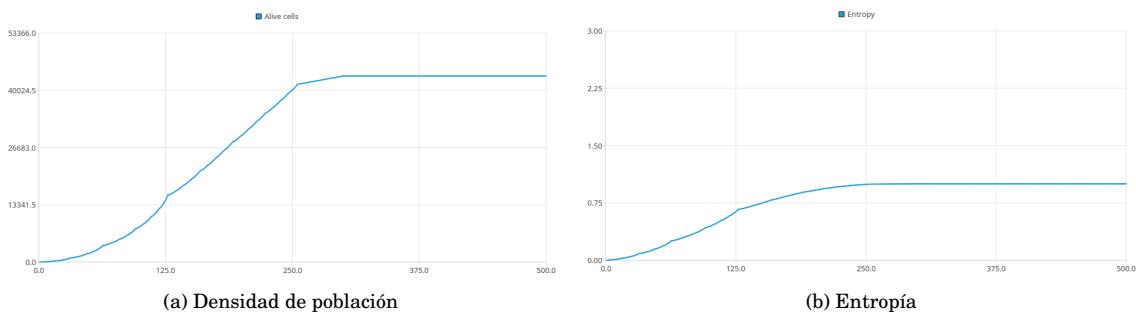


Fig. 11.2.9: Gráficas obtenidas con una sola célula viva tras 500 generaciones
Fuente: Realización propia.

Por otra parte si tenemos densidades iniciales para la configuración inicial en el rango de $(0.001, 0.999)$ el sistema no alcanza estabilidad sino que hay interacción entre las células del espacio de evolución pero existen conjuntos de células que mantienen un comportamiento aparentemente estático durante algunas generaciones hasta que alguna otra célula perturba dichos patrones y desaparece este conjunto de células, esto lo podemos observar en Fig. 11.2.10.

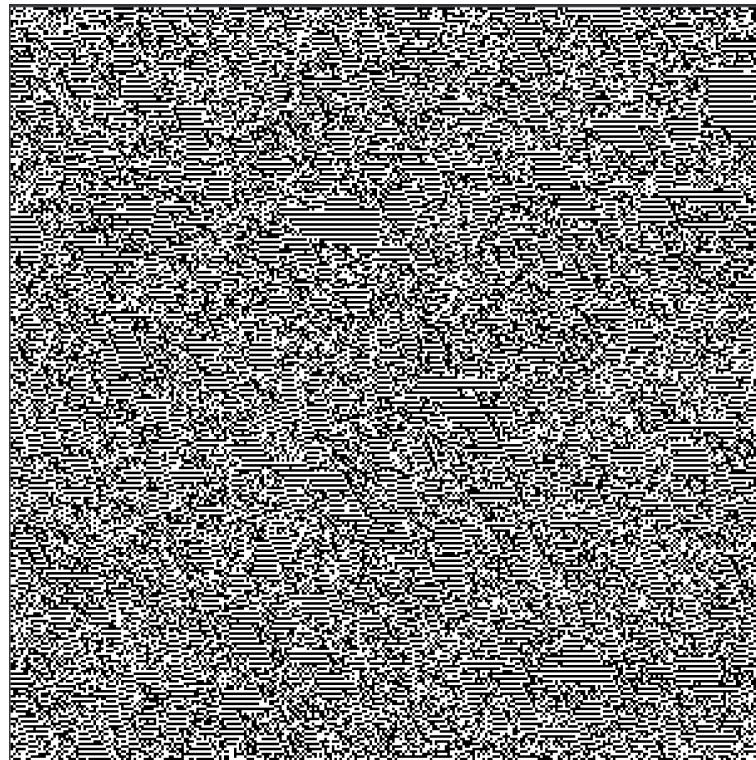


Fig. 11.2.10: Simulación de la regla 2878016454 con una densidad inicial de 5% tras 500 generaciones
Fuente: Realización propia.

En este caso es posible observar en Fig. 11.2.11 que el la densidad de población y la entropía se comportan un tanto diferente al caso anterior inicialmente, pero en la n -ésima generación los valores promedio tienden a ser similares a lo descrito en el caso anterior, en otras palabras, la densidad de población tiende a el punto fijo atractor en $(0.5, 0.5)$ mientras que la entropía tiende al valor de 1.

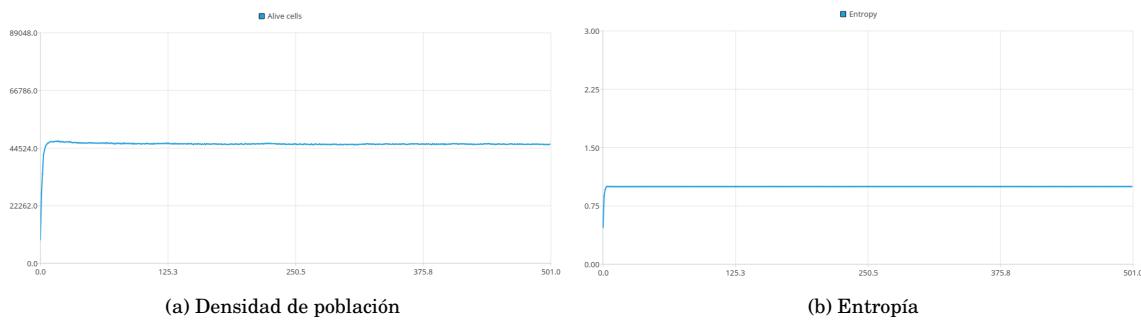


Fig. 11.2.11: Gráficas obtenidas densidad inicial del 10% tras 500 generaciones
Fuente: Realización propia.

11.3. Analizando la regla 3367827048

La regla 3367827048 se ha clasificado como del tipo complejo. Las células interactúan entre si hasta que se alcanza estabilidad en el sistema, esto ocurre si la densidad inicial es mayor o igual al 1% y menor a 100%.

Dependiendo de dicha densidad ocurren fenómenos diferentes los cuales analizaremos a continuación. El polinomio característico para esta regla es $8p^2q^3 + 4p^3q^2 + 4p^4q + p^5$ si es graficado y superpuesto con la recta identidad se obtienen cuatro puntos fijos para esta regla en particular, esto lo podemos apreciar mejor en Fig. 11.3.12.

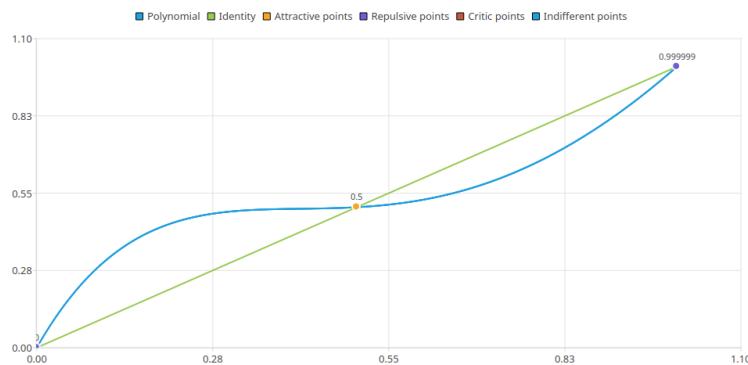


Fig. 11.3.12: Polinomio característico de la regla 3367827048

Fuente: Realización propia.

Tal y como fué mencionado anteriormente hay cuatro puntos fijos en esta regla particular, de estos existen dos del tipo atractivo, el primero en aproximadamente $(0.0, 0.0)$ y otro mas en $(0.63, 0.63)$ esto quiere decir que para densidades cercanas a cero el sistema se estabiliza a dicho punto mientras que teóricamente algo similar debe ocurrir si la densidad de la configuración inicial del sistema es cercano a 0.63 o bien es probable que en la n -ésima generación el sistema se estabilizará a dicho punto. Existen dos puntos fijos mas en $(0.22, 0.22)$ y $(0.99, 0.99)$ a diferencia de el par descrito anteriormente estos son del tipo repulsivo, en otras palabras es imposible predecir que ocurrirá con densidades cercanas a dichos puntos, tanto podrían nacer nuevas células como podrían morir algunas, se podría desatar el caos o bien estabilizarse el sistema en algún punto.

Para esta regla tenemos casos particulares, primeramente si la densidad inicial es menor o igual al 1% el sistema se mantiene estático y algunas células se mantienen vivas, en Fig. 11.3.13 se puede apreciar una configuración inicial cuya densidad es del 1%.



Fig. 11.3.13: Simulación de la regla 3367827048 con una densidad inicial de 1% tras 100 generaciones
Fuente: Realización propia.

Como podemos observar la densidad se estabiliza en aproximadamente 0.04% bastante cercano al origen, mientras que la entropía tiende a cero de igual forma. Esto lo podemos apreciar en la Fig. 11.3.14.

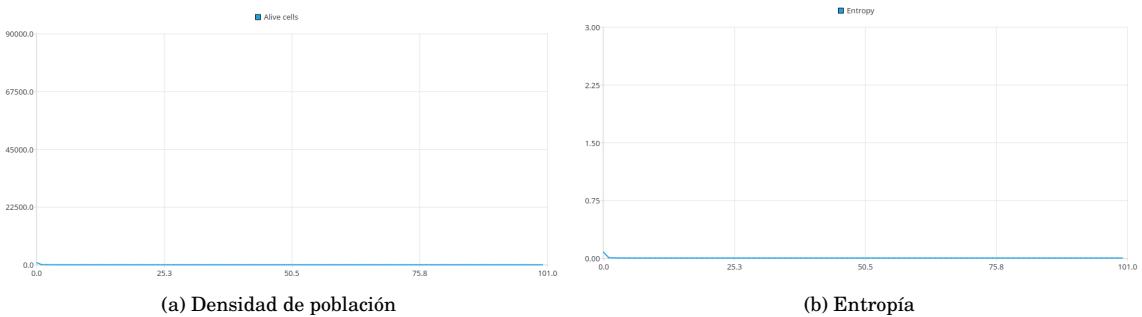


Fig. 11.3.14: Gráficas obtenidas densidad inicial del 1% tras 100 generaciones
Fuente: Realización propia.

El siguiente caso particular se presenta si la densidad es grande en concreto mayor al 99% con esta configuración solo existe desplazamiento de las células, pero no existe ningún otro movimiento o interacción entre las células del autómata celular. En Fig. 11.3.15 se muestra un ejemplo de una simulación cuya densidad inicial fué del 00%

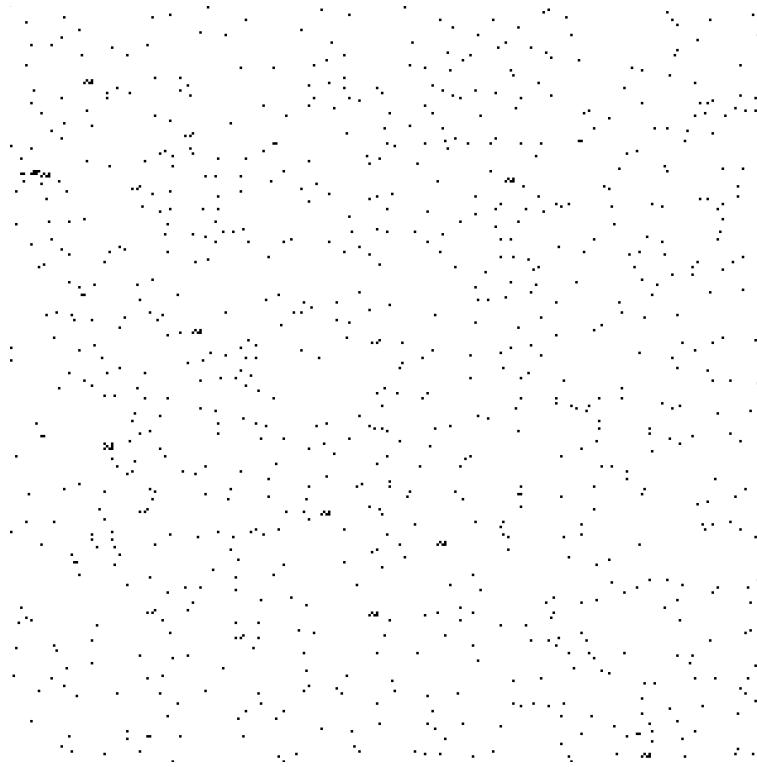


Fig. 11.3.15: Simulación de la regla 3367827048 con una densidad inicial de 99% tras 200 generaciones
Fuente: Realización propia.

A diferencia de lo ocurrido con la densidad inicial de 1% el sistema tiende a estabilizarse en valores muy cercanos a 1 y similar al caso anterior la entropía es cercana a cero tal y como se observa en Fig. 11.2.11.

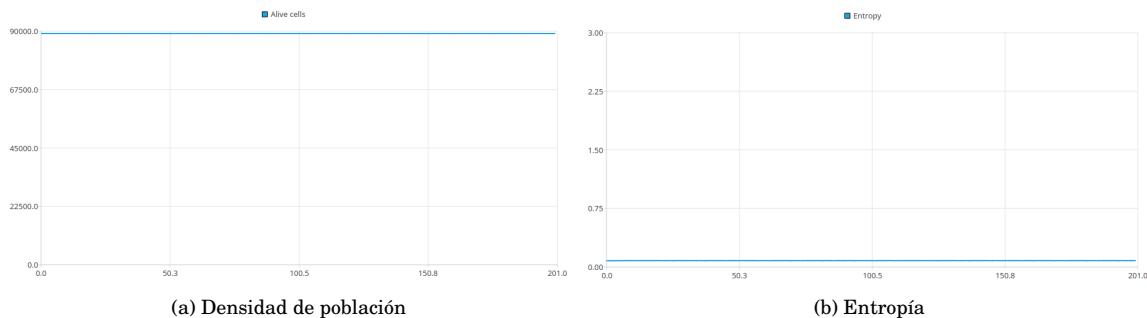


Fig. 11.3.16: Gráficas obtenidas densidad inicial del 99% tras 200 generaciones
Fuente: Realización propia.

Finalmente tenemos lo que podríamos nombrar como caso promedio, si la densidad de la configuración inicial está en el rango de (1%, 99%) se forman diversos patrones que se mantienen estáticos con el paso de las generaciones, algunos más grandes que otros pero con un desplazamiento tanto de derecha a izquierda como de izquierda a derecha, estos movimientos generan colisiones entre algunas células pero sin causar caos, sino por el contrario causan que el sistema se estabilice, como caso ejemplo se muestra la Fig. 11.3.17.

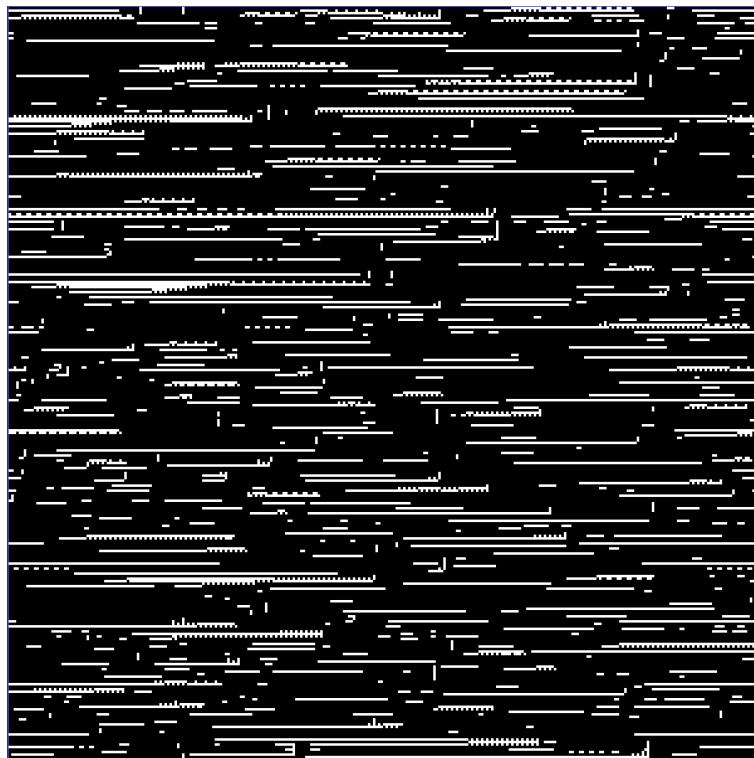


Fig. 11.3.17: Simulación de la regla 3367827048 con una densidad inicial de 50% tras 1000 generaciones
Fuente: Realización propia.

A diferencia de los casos anteriores donde las densidades en la n -ésima generación se estabilizaban en los valores límite, es decir en $(0.0, 0.0)$ o $(1.0, 1.0)$ el sistema tiende a estabilizarse cuando la densidad es cercana a 0.14 mientras que el valor de la entropía se aproxima al punto fijo en el par ordenado $(0.63, 0.63)$ tal y como se puede apreciar en Fig. 11.3.18.

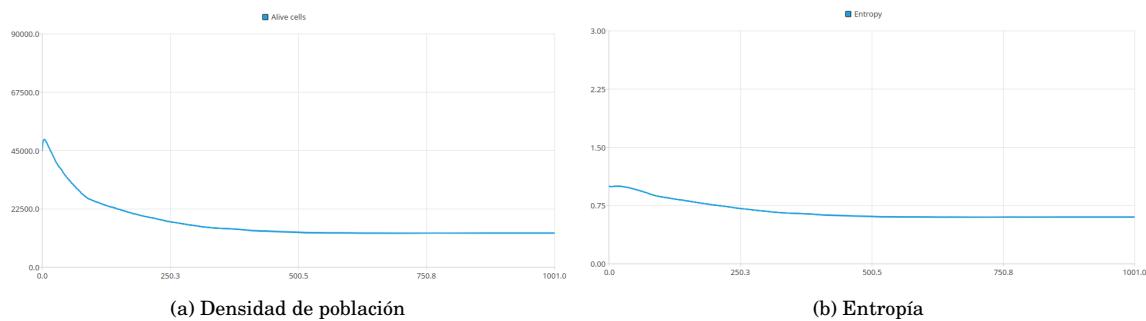


Fig. 11.3.18: Gráficas obtenidas densidad inicial del 50% tras 1000 generaciones
Fuente: Realización propia.

11.4. Analizando la regla 3740243592

Esta regla posee un comportamiento complejo interesante, debido a que las células primeramente forman patrones triangulares y adicionalmente ocurren dos tipos de movimientos. En el primero de ellos las células se desplazan hacia la derecha, mientras que en el segundo movimiento las células se desplazan hacia arriba.

En Fig. 11.4.19 muestra la gráfica del polinomio característico $f(p) = pq^4 + 6p^2q^3 + 7p^3q^2 + 4p^4q + p^5$, los puntos de intersección con la recta identidad además de la clasificación de estos puntos de intersección.

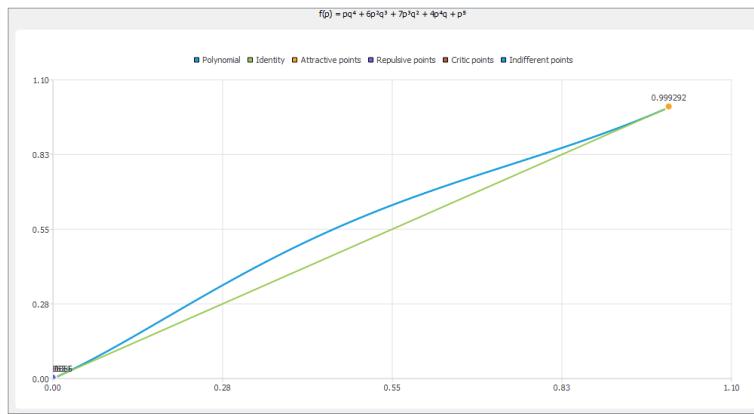


Fig. 11.4.19: Polinomio de campo promedio

Fuente: Realización propia.

Observemos el polinomio de campo promedio que es mostrado en Fig. 11.4.20 nos encontramos con dos puntos. El primer punto del tipo repulsivo se encuentra en $(0,0)$ y el segundo punto del tipo atractivo se encuentra en aproximadamente $(0.99, 0.99)$.

Utilizando la información obtenida anteriormente analicemos que ocurre si definimos densidades iniciales cercanas a dichos puntos fijos, primeramente analicemos que ocurre si la densidad inicial es muy baja (menor o igual al 5% de células vivas aproximadamente) podemos observar que se forman algunas líneas y además encontramos algunos patrones complejos que como se menciono anteriormente se desplazan en dos direcciones. En Fig. 11.4.20 podemos observar que se generan diferentes patrones y conjuntos de células.

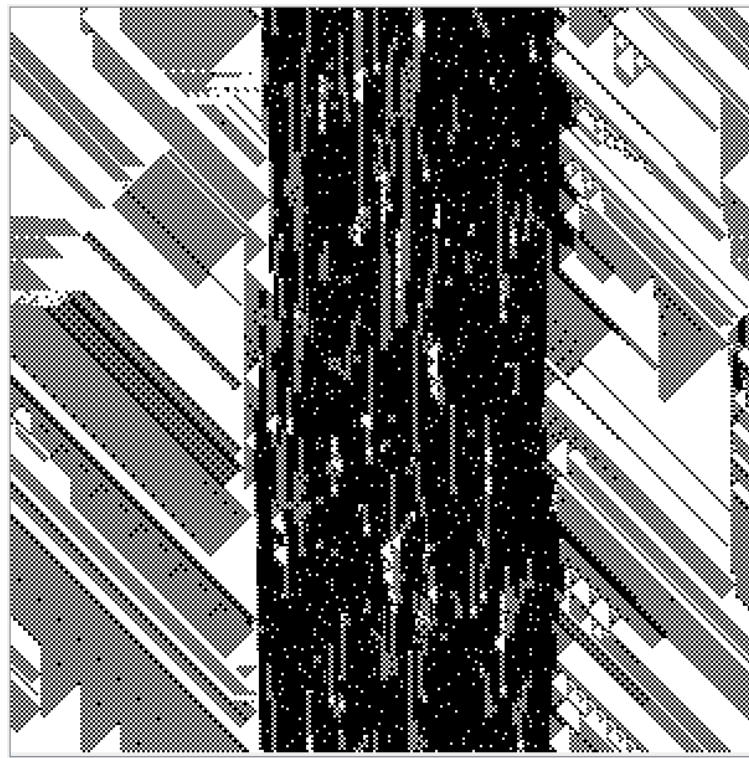


Fig. 11.4.20: Condición inicial al 5 %, generación número 1000

Fuente: Realización propia.

En Fig. 11.4.21 se muestra la gráfica de población y la gráfica de entropía.

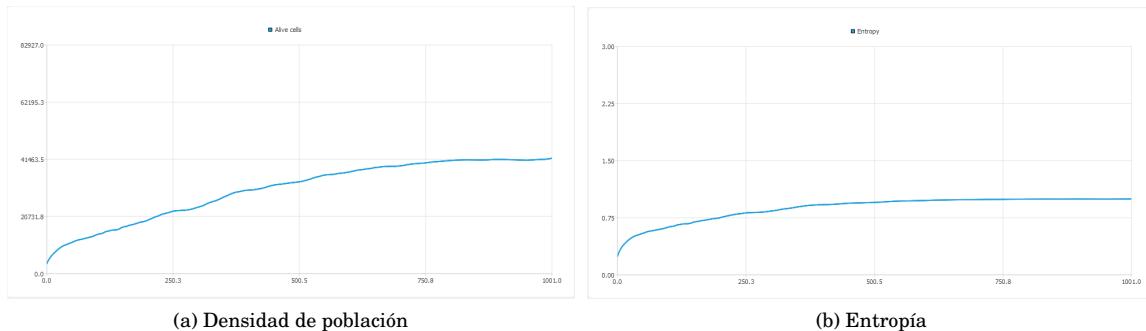


Fig. 11.4.21: Gráficas obtenidas con una densidad inicial de 5% tras 1000 generaciones

Fuente: Realización propia.

Al iniciar el autómata con una densidad inicial de células vivas mayor a 5% aproximadamente tal y como se muestra en Fig. 11.4.22 encontramos que el espacio de evolución en su mayoría aparecen patrones cuyo comportamiento es complejo además mientras mayor vaya siendo la cantidad inicial de células vivas en la n -ésima generación la densidad de células vivas se aproxima al punto fijo estable en $(0.99, 0.99)$ tal y como se muestra en Fig. 11.4.23.

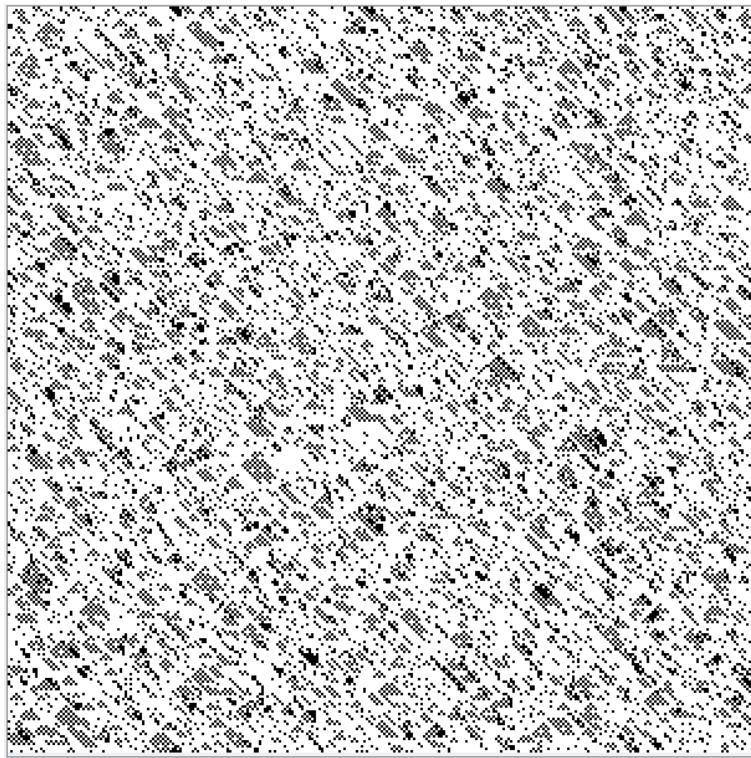


Fig. 11.4.22: Condición inicial al 5%, generación número 100

Fuente: Realización propia.

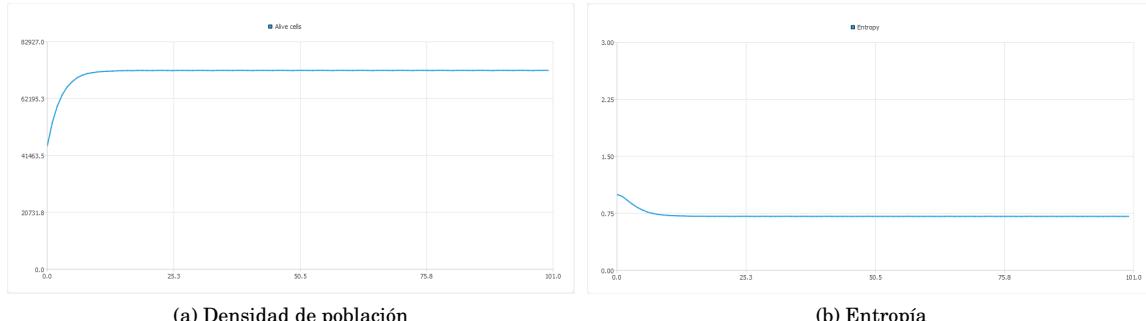


Fig. 11.4.23: Gráficas obtenidas con una densidad inicial de 50% tras 100 generaciones

Fuente: Realización propia.

11.5. Analizando la regla 4133616598

La regla 4133616598 se encuentra dentro de la clasificación de las reglas complejas. Esta regla posee un comportamiento único, en particular cuando tiene como condición inicial densidades bajas de células vivas.

En Fig. 11.5.24 se puede observar la gráfica del polinomio $f(p) = 5pq^4 + 3p^2q^3 + 9p^3q^2 + 3p^4q + p^5$

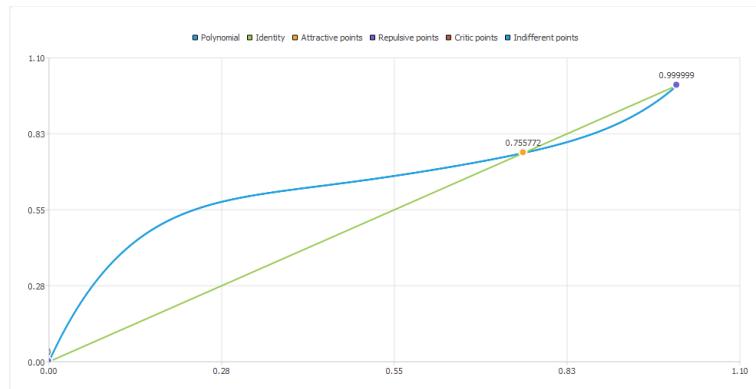


Fig. 11.5.24: Polinomio de campo promedio

Fuente: Realización propia.

Dicho polinomio presenta 3 puntos de intersección con la recta identidad. El primer punto del tipo repulsivo lo encontramos en $(0,0)$, el segundo punto de intersección del tipo atractivo se encuentra en aproximadamente $(0.75, 0.75)$, por último, el tercer punto de intersección del tipo repulsivo se encuentra en aproximadamente $(0.99, 0.99)$.

Esta regla posee la propiedad de generar patrones complejos como estructuras triangulares y ramificaciones similares a la regla 1688232692 con una única célula con vida. Los patrones triangulares se desplazan hacia la izquierda mientras que las ramificaciones aparentemente se desplazan lentamente hacia la derecha podemos observar dichos patrones en Fig. 11.5.25.



Fig. 11.5.25: Condición inicial con una célula central, generación número 500

Fuente: Realización propia.

En Fig. 11.5.26 se muestra la gráfica de densidad poblacional y la gráfica de entropía. Para la gráfica de densidad poblacional se observa un incremento de células vivas hasta alcanzar aproximadamente 70% del espacio de evolución hasta la generación 250, mientras que para la generación 500 el sistema alcanza aproximadamente 80% de células vivas cubriendo el espacio de evolución, aproximadamente en este porcentaje de células vivas el sistema se mantiene oscilando para las generaciones posteriores.

Para la gráfica de entropía se observa un incremento hasta la generación 200 aproximadamente, después de esta generación la cantidad de entropía comienza a disminuir hasta la generación 250 aproximadamente, posteriormente, el nivel de entropía se mantiene casi constante para las siguientes generaciones.

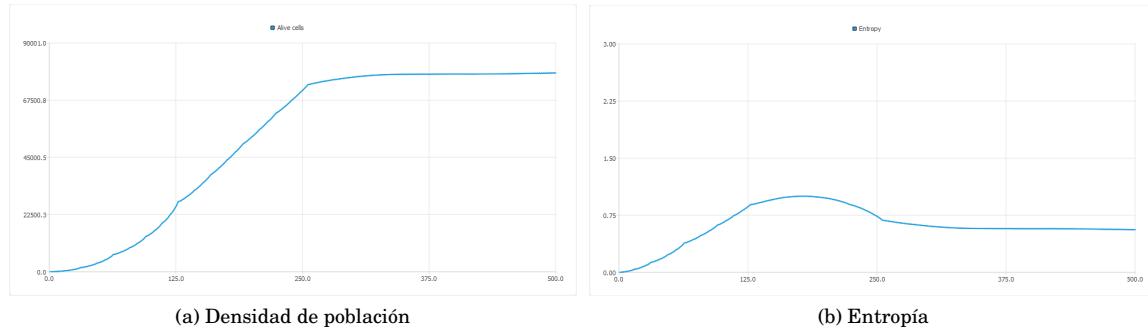


Fig. 11.5.26: Gráficas obtenidas con una densidad inicial de una célula tras 500 generaciones
Fuente: Realización propia.

Al iniciar el autómata con una cantidad pequeña de células vivas la densidad crece rápidamente, ya no se perciben con facilidad los patrones triangulares y las ramificaciones que eran posible observar en el caso anterior, sino que únicamente se aprecian observando cuidadosamente los movimientos que generan las células con el estado 1, como ejemplo tenemos lo que se muestra en Fig. 11.5.27.

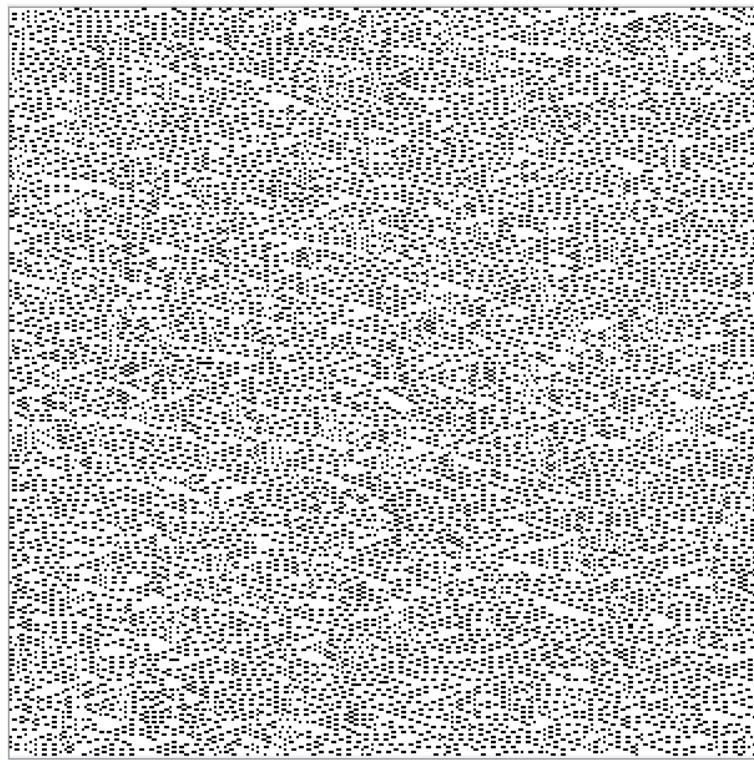


Fig. 11.5.27: Condición inicial al 5%, generación número 50

Fuente: Realización propia.

A continuación se presenta la gráfica de población y la gráfica de entropía Fig. 11.5.28. Nos encontramos con dos gráficas muy similares al caso particular donde solo hay una célula viva. La cantidad de células incrementa a muy altas densidades mientras que la entropía incrementa por algunas generaciones y finalmente desciende para estabilizarse para las generaciones siguientes.

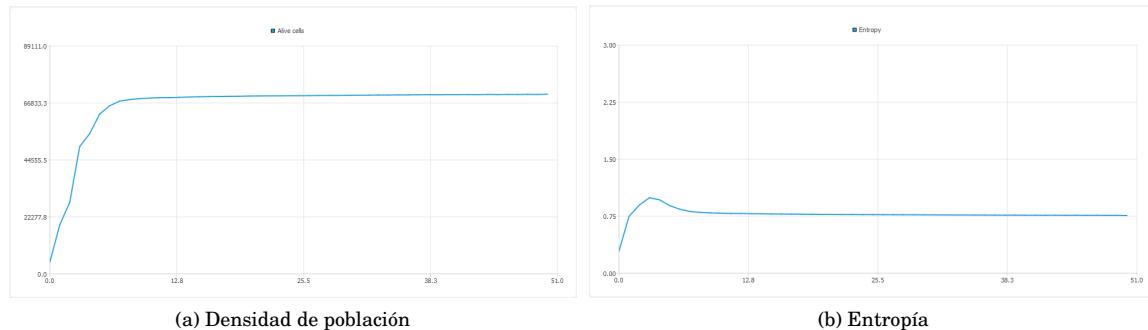


Fig. 11.5.28: Gráficas obtenidas con una densidad inicial del 5% tras 50 generaciones

Fuente: Realización propia.

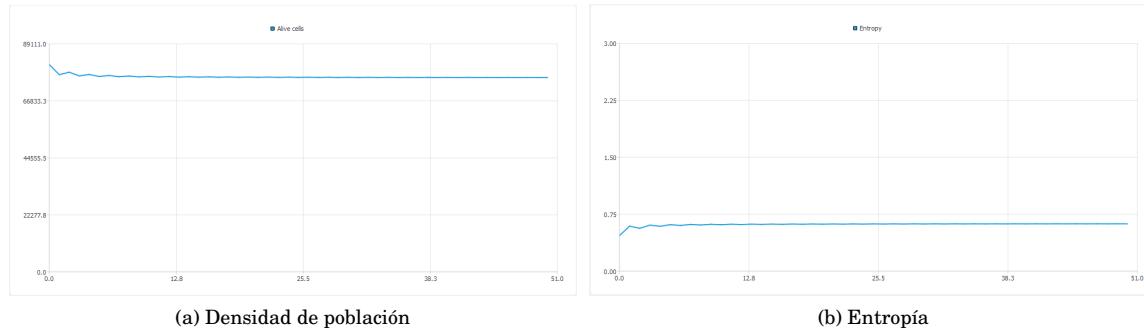
Cuando iniciamos el autómata con una cantidad alta de células la variación que existe es muy poca en comparación con densidades iniciales pequeñas tal y como se muestra en Fig. 11.5.29.



Fig. 11.5.29: Condición inicial al 90%, generación número 50

Fuente: Realización propia.

La densidad poblacional no tiene una gran variación si es comparada con las configuraciones anteriores, debido a que con el 90% de población inicial el sistema se estabiliza cuando alcanza aproximadamente 84% de densidad, mientras que en configuraciones iniciales menores a 90% la densidad poblacional alcanza estabilidad entre 78% y 84%.



(a) Densidad de población

(b) Entropía

Fig. 11.5.30: Gráficas obtenidas con una densidad inicial del 90% tras 50 generaciones

Fuente: Realización propia.

La gráfica de densidad poblacional muestra como la población disminuye muy poco y en pocas generaciones encuentra la estabilidad. La gráfica de entropía muestra un comportamiento similar, oscila un poco en las primeras generaciones hasta que encuentra una estabilidad esto se aprecia en Fig. 11.5.30.

11.6. Analizando la regla 3185432644

La regla 3185432644 clasificada como compleja, posee la característica de tener dos tipos de movimientos al igual que la regla 3740243592. Los dos tipos de movimiento que presenta esta regla es con dirección hacia abajo y con dirección hacia la derecha.

En Fig. 11.6.31 se muestra la gráfica del polinomio $f(p) = 2pq^4 + 5p^2q^3 + 6p^3q^2 + 4p^4q + p^5$.

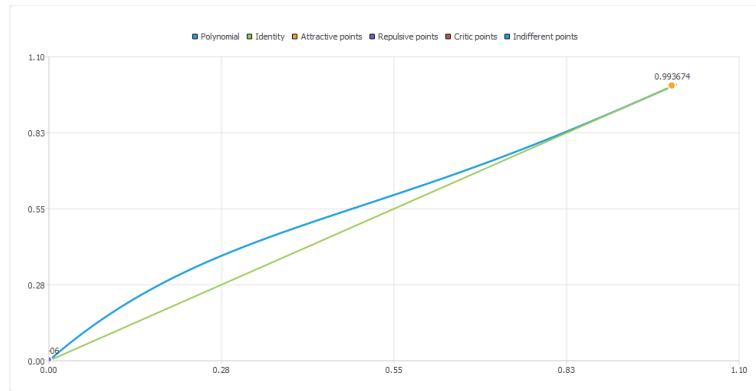


Fig. 11.6.31: Polinomio de campo promedio

Fuente: Realización propia.

Podemos observar que existen dos puntos de intersección entre el polinomio característico y la recta identidad, el primero de ellos se encuentra en aproximadamente $(0,0)$ del tipo repulsivo, mientras que el segundo punto del tipo atractivo lo encontramos en aproximadamente $(0.99, 0.99)$.

Cuando se inicia el autómata con densidades iniciales pequeñas, es decir, menores o igual al 0.1% tenemos que las células forman líneas diagonales con pendiente negativa como se muestra en Fig. 11.6.32.

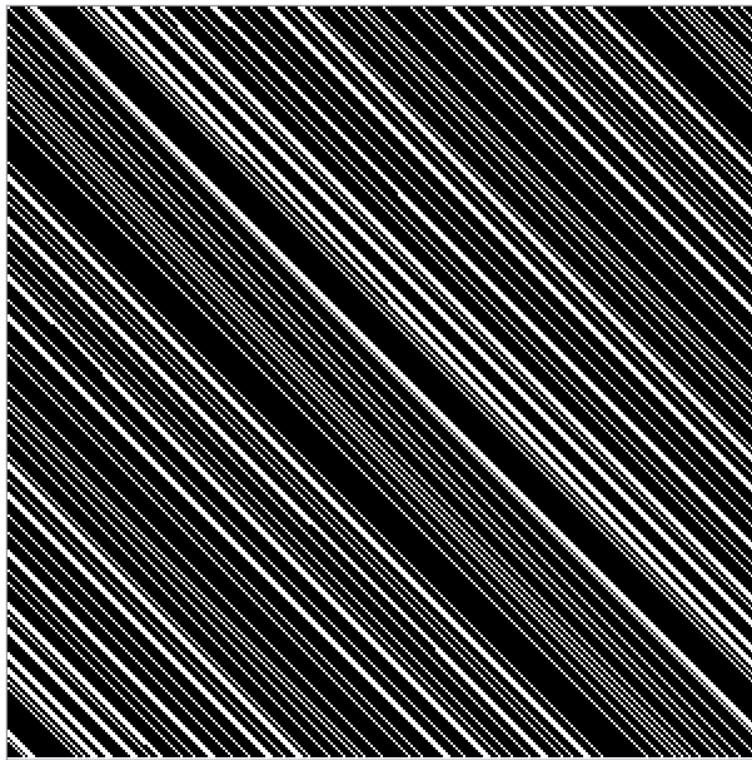
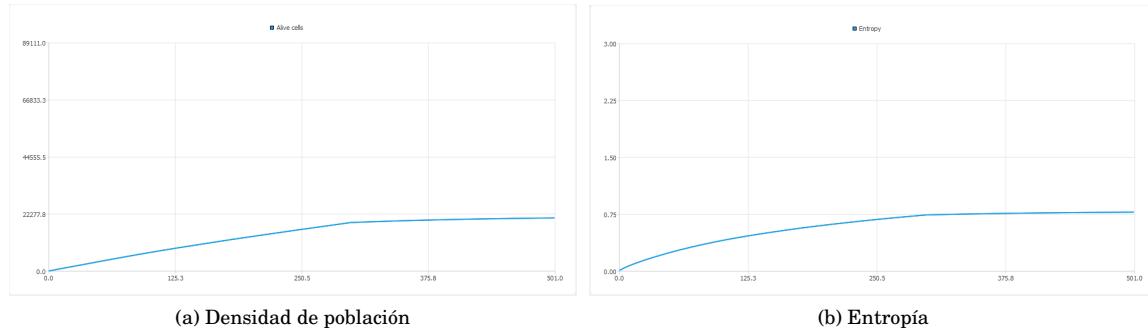


Fig. 11.6.32: Condición inicial al 0.1%, generación número 500

Fuente: Realización propia.

A continuación se muestra en Fig. 11.6.33 la gráfica de densidad poblacional y la gráfica de entropía correspondiente a la configuración anteriormente mencionada.

Fig. 11.6.33: Gráficas obtenidas con una densidad inicial del 0.1% tras 500 generaciones
Fuente: Realización propia.

Al establecer una densidad inicial del autómata alta se aprecian algunos patrones triangulares que se desplazan de izquierda a derecha, además de otras formaciones que se desplazan de arriba hacia abajo como se puede ver en Fig. 11.6.34.

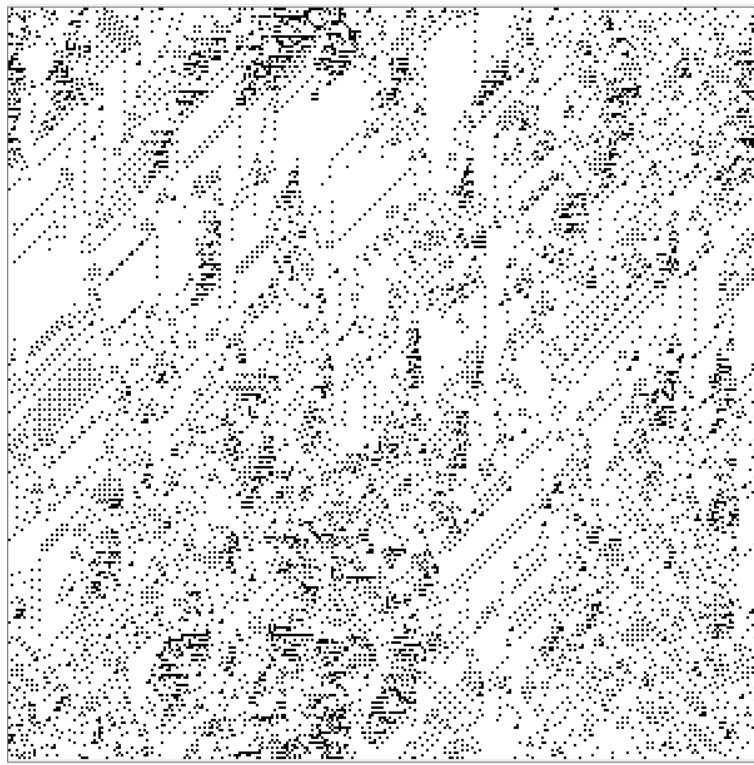
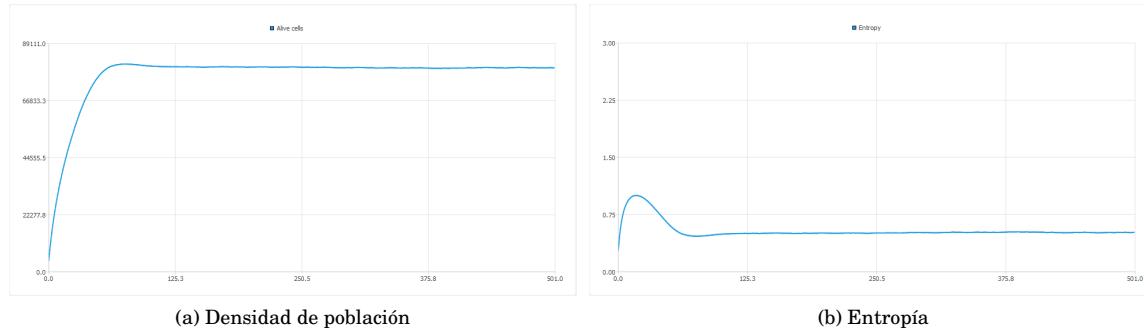


Fig. 11.6.34: Condición inicial al 5%, generación número 500

Fuente: Realización propia.

En la Fig. 11.6.35 se aprecian conjuntos de células vivas, estas estructuras se desplazan hacia abajo, por otro lado también se observan pequeñas líneas horizontales de células cuyo estado es cero, estas formaciones se desplazan hacia la izquierda, en ocasiones se pueden observar pequeños patrones triangulares que de igual forma se desplazan hacia la derecha.



(a) Densidad de población

(b) Entropía

Fig. 11.6.35: Gráficas obtenidas con una densidad inicial del 5% tras 500 generaciones

Fuente: Realización propia.

En la Fig. 11.6.35a de densidad poblacional se observa un incremento de células en pocas generaciones para después alcanzar estabilidad en el sistema, algo similar ocurre con la Fig. 11.6.35b de entropía, mientras la densidad poblacional incrementa el nivel de entropía también lo hace, cuando la densidad poblacional se estabiliza el nivel de entropía disminuye hasta que el sistema encuentra estabilidad.

Para una configuración inicial con una densidad tan alta como 80% el sistema encuentra una estabilidad demasiado pronto como se muestra en Fig. 11.6.36.

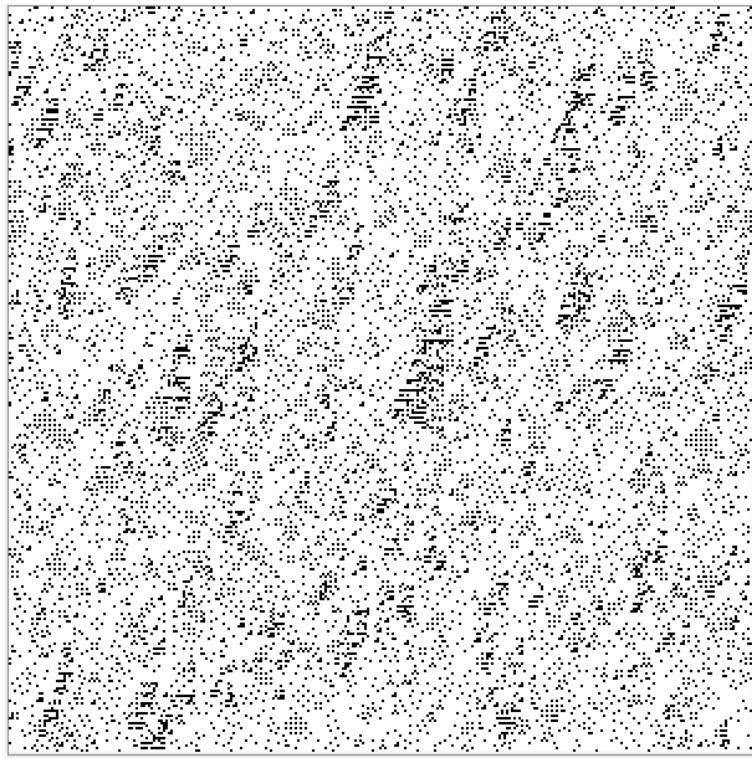


Fig. 11.6.36: Condición inicial al 80 %, generación número 500

Fuente: Realización propia.

En Fig. 11.6.37 se aprecian con dificultad algunos patrones que se desplazan hacia la derecha, aunque en su mayoría el espacio de evolución se encuentra conformado por células con vida que se desplazan hacia abajo.

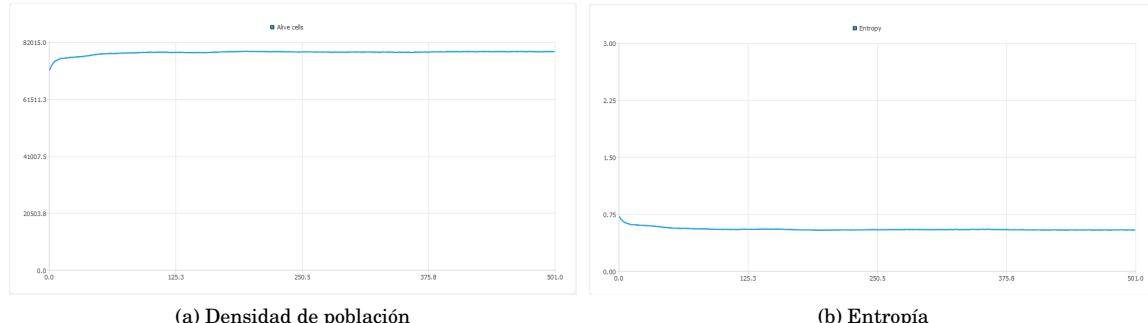


Fig. 11.6.37: Gráficas obtenidas con una densidad inicial del 80% tras 500 generaciones

Fuente: Realización propia.

En la Fig. 11.6.37a de densidad de población y la Fig. 11.6.37b de entropía se aprecia como el sistema encuentra estabilidad muy rápido, de hecho, las gráficas se encuentran casi constantes a través de las generaciones. Para apreciar con claridad el comportamiento de esta regla es recomendable iniciar el sistema con densidades bajas de células vivas.

A esta regla se le pueden atribuir distintos comportamientos y distintos patrones como se mencionó anteriormente. Entre los patrones más interesantes se pueden encontrar *gliders*, es decir, conjuntos de células que se desplazan a lo largo del espacio de evoluciones.

A continuación se presentan imágenes de algunos de los gliders más usuales que se pueden encontrar.

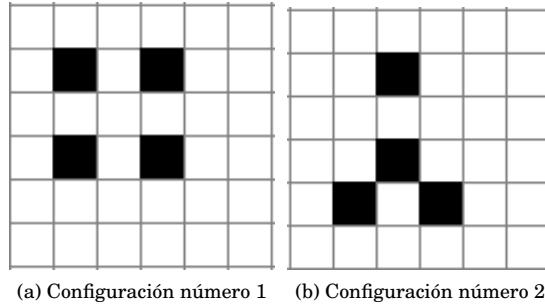


Fig. 11.6.38: Glider con 2 configuraciones

Fuente: Realización propia.

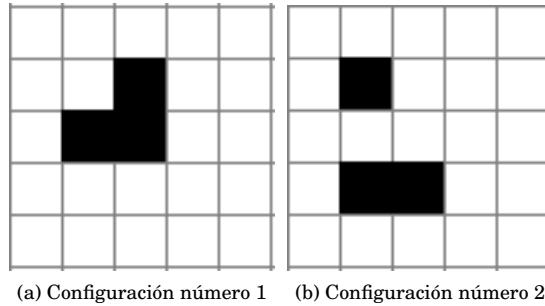


Fig. 11.6.39: Glider con 2 configuraciones

Fuente: Realización propia.

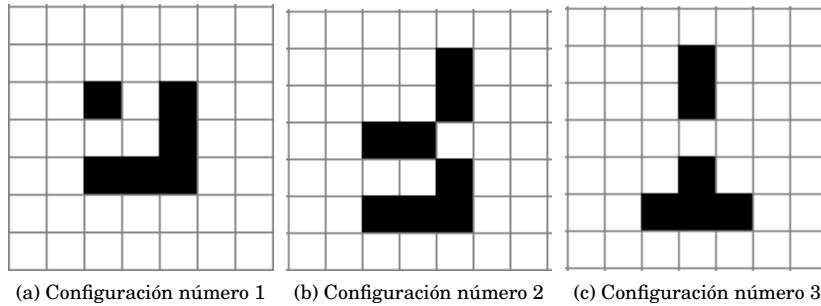


Fig. 11.6.40: Glider con 3 configuraciones

Fuente: Realización propia.

Generalmente estos gliders se desplazan de arriba hacia abajo y en ocasiones, cuando algunos de estos gliders colisionan se generen nuevos patrones como se muestra en Fig. 11.6.41a.

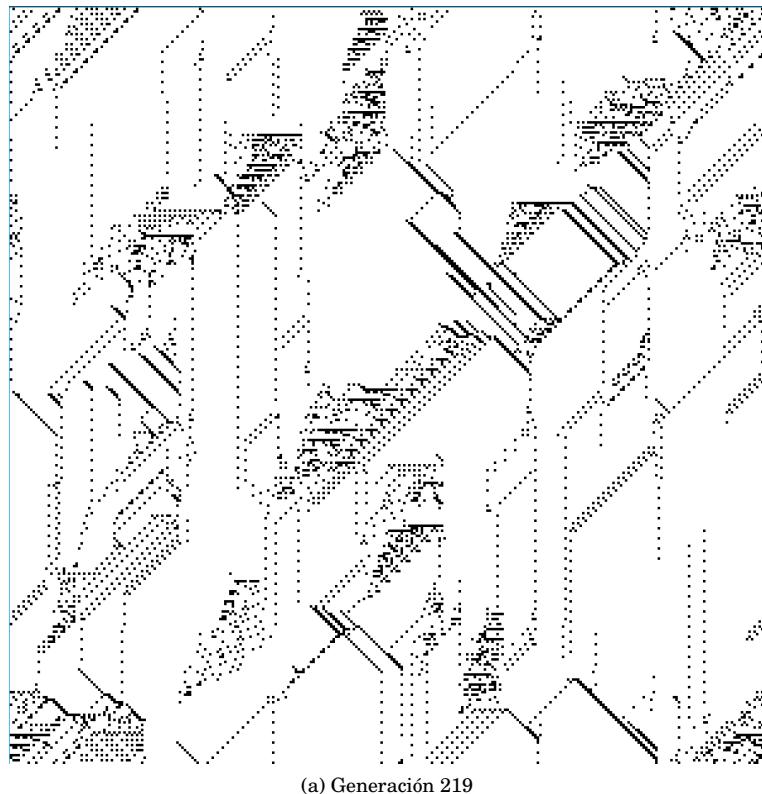


Fig. 11.6.41: Colisión de gliders

Fuente: Realización propia.

12

Conclusiones

12.1. Resultados

La investigación realizada como parte de este trabajo permite observar la importancia de explorar a detalle un AC que puede parecer simple con su configuración inicial predeterminada, pero al modificar ya sea el tipo de regla (totalista o semitotalista) e incluso el tipo de vecindad aparecerán patrones y comportamientos que resultan muy interesantes como los que se observan en los casos de estudio donde se encontraron *gliders*, el caso específico de la regla 1688232692 que tiene patrones que se asemejan con la corteza de un árbol, y muestra que este trabajo puede tener aplicaciones más allá del campo de sistemas computacionales. Incluso, se encontraron patrones que forman fractales con diferentes desplazamientos, formas e incluso comportamientos, mismos que son estudiados en el campo de las ciencias exactas y tal como se observa en el estado del arte, pueden tener aplicaciones en el campo de la criptografía.

Este trabajo tiene como propósito no solo explorar las diferentes reglas, sino que además realizar un análisis estadístico mediante la teoría del campo promedio que nos permite obtener un polinomio característico para cada una de ellas, así como observar los cambios con diferentes densidades. Por ello, el software desarrollado a la medida fue de gran ayuda, primeramente porque contribuyó a la verificación de cálculos de una forma más rápida y precisa que si se realizaran de forma manual, seguido de su aporte visual tanto para las gráficas del polinomio característico y sus intersecciones con la recta identidad, de la entropía y de la densidad de las células conforme al paso de las generaciones, así como para el espacio de evoluciones contribuyendo al descubrimiento de patrones en estos universos discretos que se exploraron permitiéndonos corroborar la clase a la que pertenece cada uno según sus características morfológicas y fenotípicas. Por lo tanto, se ratifica que los requerimientos obtenidos y propuestos fueron suficientes para realizar el estudio y exploración de las reglas, principalmente bajo la vecindad de von Neumann en un espacio cerrado utilizando las reglas totalistas.

Recapitulando y a manera de conclusión se puede mencionar que el estudio de las reglas totalistas bajo la vecindad de von Neumann ha arrojado resultados satisfactorios y relevantes debido a las clases y patrones que se encontraron, ya que siguiendo la clasificación propuesta por Stephen Wolfram para autómatas celulares unidimensionales se pudieron apreciar diferentes ejemplos de cada clase trasladando la misma teoría a los AC en dos dimensiones mediante el uso del sistema vNCASimulator . Por ejemplo, para la clase I se encontró que la regla 4293902335 tiene un solo punto atractivo en (0,0) y permite asociarla con una *clase I*, misma que se verifica porque evoluciona a una configuración estable y homogénea, la regla 1077936134 tiene un par de puntos, del tipo repulsivo en (0,0) y del tipo atractivo en (0.16,0.16)y se puede asociar con una *clase II* y se verifica observando su evolución que es periódica constante, la regla 1688232692 que es de *clase IV* debido a su comportamiento complejo cuyo polinomio tiene dos puntos fijos, del tipo repulsivo en (0,0) y del tipo atractivo en (0.34,0.34). Con esto se constata que el cálculo del polinomio aplicando la teoría del campo promedio da una pauta para clasificar las reglas de acuerdo con las propiedades estadísticas. El sistema vNCASimulator además permitió ver más allá de lo que puede significar la modificación de una configuración inicial y sus repercusiones en un sistema dinámico específico como lo son los AC y que este mismo trabajo puede extenderse

bastante y contribuir al entendimiento de casos reales por asociación y semejanza a varios campos tanto de las ciencias sociales como exactas.

12.2. Limitaciones

Esta investigación tiene ciertas limitaciones que deben ser mencionadas y discutidas. La exploración de reglas se hace mediante una configuración manual o aleatoria y se puede considerar un problema tanto para la vecindad de von Neumann como la de Moore debido a la cantidad de reglas disponibles, por lo cual se podría automatizar para explorar todas las combinaciones posibles mediante un algoritmo genético o haciendo uso de inteligencia artificial, por mencionar algunos.

También se encontró que el uso de seis decimales al momento de evaluar el polinomio característico previamente obtenido mediante la teoría del campo promedio pueden no ser suficientes, por lo que se deberá aumentar el número de los mismos lo cual tendrá un impacto en la precisión y así tener resultados más exactos.

Además, las investigaciones realizadas en este tipo de espacios siguen siendo escasas, por lo que proponer una clasificación definitiva sigue siendo un tema abierto que requiere examinar a mayor profundidad en fundamentos matemáticos, así como el conocimiento de todas las características fenotípicas y morfológicas aplicables para autómatas en dos dimensiones.

12.3. Trabajo por realizar

Dadas las limitaciones encontradas a lo largo de la realización del trabajo se puede observar que hay módulos que se pueden agregar o mejorar para obtener mejores resultados que contribuyan al mismo, así como la continuación de la exploración y documentación de resultados interesantes.

Para el simulador queda como trabajo a futuro realizar un análisis con diferentes tipos de entropía además de la de Shannon para reafirmar los resultados obtenidos, lo cual requiere de una previa y profunda investigación para determinar cual es la más adecuada. El gradiente de colores para las células vivas es fijo, por lo cual al llegar a un cierto color entra a un bucle que se puede modificar para poder visualizar mejor las células que se mantienen vivas a lo largo de la simulación y tener un mayor espectro de referencia. El simulador tiene un módulo que permite realizar una proyección a tres dimensiones, el cual se puede mejorar para que se distinga mejor cada generación que ha transcurrido a lo largo de la simulación.

En cuestiones de investigación y exploración se puede extender el estudio de los puntos fijos aumentando el número de decimales a utilizar para hacer un cálculo más preciso en las intersecciones del polinomio con la recta identidad, realizar una exploración más extensa sobre las reglas totalistas y ampliar el estudio con reglas semitotalistas para descubrir patrones y comportamientos que resulten interesantes y clasificarlos.

Finalmente, para realizar una exploración más rápida se puede desarrollar un segundo programa que automatice el cálculo de puntos fijos para las millones de reglas existentes, principalmente para la vecindad de von Neumann y de Moore y las clasifique guiado por el trabajo presentado en este documento. Así mismo, se pueden implementar otros métodos numéricos que ayuden a corroborar los datos obtenidos y/u optimizar los cálculos que se realicen para hacerlo más preciso.

Bibliografía

- [1] P. A. León y R. Basurto, *Introducción a los Sistemas Dinámicos y Autómatas Celulares*, Trabajo de Investigación. Escuela Superior de Cómputo. Instituto Politécnico Nacional. México, 2008.
- [2] *Matemáticas para los Modelos Dinámicos*, notas de clase para MATECOEMPR, Departamento de Matemáticas para la Economía y la Empresa, Facultad de Economía. 2019.
- [3] E. A. Lacomba, “Los sistemas dinámicos, ¿Qué son y para qué sirven?”, *Miscelánea Matemática*, no. 32, pp. 42-48, noviembre, 2000. [En línea]. Disponible en <http://albertofest.matcuer.unam.mx/Misc32/Lacomba.pdf>
- [4] A. Garfinkel, J. Shevtsov e Y. Guo, “Modeling Life. The Mathematics of Biological Systems”. Los Angeles, CA. Springer, 2017, pp. 361-370. doi: <https://doi.org/10.1007/978-3-319-59731-7>
- [5] M. Loaiza Ramírez, “Diseño y simulación de un criptosistema caótico para comunicaciones seguras”. Tesis, Licenciatura en Ingeniería en Electrónica y Comunicaciones. Departamento de Computación, Electrónica, Física e Innovación. Escuela de Ingeniería y Ciencias, Universidad de las Américas Puebla, México, 2006. pp. 26-27. Disponible en: http://catarina.udlap.mx/u_dl_a/tales/documentos/1em/loaiza_r_m/capitulo3.pdf
- [6] C. Oestreicher, “A history of chaos theory,” *Dialogues in clinical neuroscience*, vol. 9, no. 3, pp.279-289, sep. 2007. PMID: 17969865; PMCID: PMC3202497.
- [7] R. Bishop, “Chaos”, *The Stanford Encyclopedia of Philosophy*, Edward N. Zalta (ed.), 2017. [En línea]. Disponible en: <https://plato.stanford.edu/entries/chaos/>
- [8] H. L. Poe y J. H. Davis, “When Is Order Disorder?”, *Science and Faith: An Evangelical Dialogue*, Broadman and Holman Publishers, pp. 201-214, 2000. [En línea]. Disponible en: <https://www.uu.edu/societies/inklings/books/scienceandfaith/Chapter13.pdf>
- [9] C. J. Martínez Moncaleano, “Teoría del caos y estrategia empresarial”. *Revista de la Facultad de Ciencias Económicas y Administrativas*, vol. 19, no. 1, pp. 204-214, junio 2018. doi: <http://dx.doi.org/10.22267/rtend.181901.94>
- [10] G. A. Laguna Sánchez, R. M. Jiménez, G. A. Patrick Encina y G. Vázquez Hernández, “Complejidad y sistemas complejos: un acercamiento multidimensional”, CopIt-arXives, 2016. p. 1. [En línea] Disponible en: <http://scifunam.fisica.unam.mx/mir/copit/TS0013ES/TS0013ES.pdf>
- [11] Y. Bar-Yam, “Dynamics of complex systems”. Addison-Wesley, 1997, USA, pp. 1-13.

- [12] J. Ladyman, J. Lambert y K. Wiesner, "What is a complex system?", *European Journal for Philosophy of Science*, vol. 3, pp. 33-67, mayo, 2013. doi: 10.1007/s13194-012-0056-8
- [13] "¡Emergencia, Emergencia!", *Investigación y Ciencia*, febrero, 2014. [En línea]. Disponible en: <https://www.investigacionyciencia.es/blogs/fisica-y-quimica/34/posts/emergencia-emergencia-11817>
- [14] M. Vivanco, "Emergencia. Concepto Y Método". Departamento de Sociología, Universidad de Chile, Santiago, Chile, 2014. [En línea]. doi: <http://dx.doi.org/10.4067/S0717-554X2014000100004>
- [15] S. Wolfram, "The Crucial Experiment", *A new kind of science*, 2002, cap. 2, sec. 3, pp. 42-50. [En línea]. Disponible en: <https://www.wolframscience.com/nks/>
- [16] S. Wolfram, "Cellular Automata", *Cellular Automata and Complexity: collected papers*, Westview Press, 1994, cap. 2, pp. 412-413. [En línea]. Disponible en: <https://www.stephenwolfram.com/publications/cellular-automata-complexity/pdfs/cellular-automata.pdf>
- [17] J. M. Gómez Soto, *Enfoques de estudio en los Autómatas Celulares Lineales*, Trabajo de Investigación. Departamento de Ingeniería Eléctrica, Sección de Computación. Centro de Estudios Avanzados del Instituto Politécnico Nacional. Agosto, 1996. México
- [18] S. Wolfram, "Starting from Randomness", *A new kind of science*, 2002, cap. 6, sec. 2, pp. 231-249. [En línea]. Disponible en: <https://www.wolframscience.com/nks/>
- [19] G. Juárez Martínez, "Teoría del Campo Promedio en Autómatas Celulares similares a 'The Game Of Life'", Tesis. Maestro en Ciencias con especialidad en Ingeniería Eléctrica. Centro de Investigación y de Estudios Avanzados del I.P.N. Noviembre, 2000. [En línea]. Disponible en: http://www.comunidad.escom.ipn.mx/genaro/Papers/Veranos_McIntosh_files/Articulo%20Verano%20De%20Investigacion%202011.pdf
- [20] R. J. Krawczyk, "Architectural Interpretation of Cellular Automata". College of Architecture, Illinois Institute of Technology, Chicago, 2002. [En línea]. Disponible en: <https://mypages.iit.edu/~krawczyk/rjkga02.pdf>
- [21] U. Roncoroni, "L-Systems Generator". Junio, 2007. [En línea]. Disponible en: <http://www.digitalpoiesis.org/gallery.html>
- [22] D. A. Reyes Gómez, "Descripción y Aplicaciones de los Autómatas Celulares", Trabajo de Investigación. Departamento de Aplicación de Microcomputadoras Universidad Autónoma de Puebla. México, 2011.
- [23] A. Cano Rojas, Á. Rojas Matas. "Autómatas celulares y aplicaciones", *Revista Iberoamericana de Educación Matemática*, no. 46, pp. 33-48, junio 2016. [En línea]. Disponible en: http://www.fisem.org/www/union/revistas/2016/46/01_13-307-1-ED.pdf
- [24] S. C. Fu, G. Milne, *Epidemic Modelling Using Cellular Automata*, Trabajo de Investigación. School of Computer Science and Software Engineering. 2003, Canberra, Australia. [En línea]. Disponible en <https://smr.csse.uwa.edu.au/pdf/EpidemicModellingUsingCA.pdf>
- [25] H. Gutowitz, C. Langton, "Mean Field Theory of the Edge of Chaos", en *Advances in Artificial Life*, Granada, España, 1995, pp. 1-4. doi: https://doi.org/10.1007/3-540-59496-5_288
- [26] N. H. Packard, S. Wolfram, "Two Dimensional Cellular Automata", *Journal of Statistical Physics*, vol. 38, nos. 5-6, pp. 902-903, marzo, 1985. doi: <https://doi.org/10.1007/BF01010423>
- [27] A. Amon, "Nonlinear dynamics". *Phénomènes nonlinéaires et chaos*. Francia, 2007. [En línea]. Disponible en: <https://hal.archives-ouvertes.fr/cel-01510146v2/document>

- [28] D. C. Carmona Collado, "Fractal Attraction". Tesis profesional. Licenciatura en Artes Plásticas. Departamento de Artes Plásticas y Teatro. Universidad de las Américas Puebla. Cholula, Puebla, México. 2003, pp. 1-3. [En línea]. Disponible en: http://catarina.udlap.mx/u_dl_a/tales/documentos/lap/carmona_c_dc/capitulo1.pdf
- [29] A. Prieto Guerrero, G. Espinosa Paredes, "Nonlinear signal processing methods: DR estimation and nonlinear stability indicators", *Linear and Non-Linear Stability Analysis in Boiling Water Reactors*, Woodhead Publishing Series in Energy, 2019, cap. 7, sec. 4.2, pp. 315-398. doi: <https://doi.org/10.1016/B978-0-08-102445-4.00007-2>
- [30] K. Krippendorff, "Mathematical Theory of Communication", *Encyclopedia of Communication Theory*. Sage Publications, Los Angeles, CA. 2009, pp. 614-618. [En línea]. Disponible en: https://repository.upenn.edu/cgi/viewcontent.cgi?article=1172&context=asc_papers
- [31] C.E. Shannon, "A Mathematical Theory of Communication", *The Bell System Technical Journal*, vol. 27, no.3, pp. 379-423, octubre, 1948. [En línea]. Disponible en: <http://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf>
- [32] G. Juárez Martínez, "ACOSXL21 para Mac OS X". Agosto, 2001. [En línea]. Disponible en: <http://delta.cs.cinvestav.mx/~mcintosh/comun/tesismaestria/genaro/node43.html>
- [33] G. Juárez Martínez, "ACOSXSTV para Mac OS X". Agosto, 2001. [En línea]. Disponible en: <http://delta.cs.cinvestav.mx/~mcintosh/comun/tesismaestria/genaro/node44.html>
- [34] G. Juárez Martínez, "ACOSXSTM para Windows NT". Agosto, 2001. [En línea]. Disponible en: <http://delta.cs.cinvestav.mx/~mcintosh/comun/tesismaestria/genaro/node45.html>
- [35] G. Juárez Martínez, "ACOSXV para OpenStep". Agosto, 2001. [En línea]. Disponible en: <http://delta.cs.cinvestav.mx/~mcintosh/comun/tesismaestria/genaro/node46.html>
- [36] G. Juárez Martínez, "ACOSXM para Mac OS X Server". Agosto, 2001. [En línea]. Disponible en: <http://delta.cs.cinvestav.mx/~mcintosh/comun/tesismaestria/genaro/node47.html>
- [37] R. Campaña, "El proceso de desarrollo rápido de aplicaciones (DRA) de software: Un aporte práctico en el Instituto Geográfico Militar". Gestión de Investigación y Desarrollo, Instituto Geográfico Militar. Quito, Ecuador, marzo, 2015. [En línea]. Disponible en: https://www.researchgate.net/publication/303839299_El_proceso_de_desarrollo_rapido_de_aplicaciones_DRA_de_software_Un_aporte_practico_en_el_Instituto_Geografico_Militar/
- [38] P. Beynon-Davies, C. Carne, H. Mackay, D. Tudhope, "Rapid application development (RAD): An empirical review". *European Journal of Information Systems*, pp. 221-223 Diciembre, 2017. doi: <https://doi.org/10.1057/palgrave.ejis.3000325>
- [39] I. Sommerville, "Requerimientos del software". *Ingeniería del software*. Séptima edición. Pearson Educación, S.A. Madrid, 2005. [En línea]. Disponible en: http://zeus.inf.ucv.cl/~bcrawford/AULA_ICI_3242/IngenieriadelsSoftware7ma.Ed.-IanSommerville.pdf