Modelación y simulación computacional basada en Agentes

Práctica 1: Implementación y análisis de autómatas celulares

Alejandro Terrazas Rivera 421006692
Ángela Janín Ángeles Martínez 314201009

Parte 1. Preguntas.

1) ¿Qué es la modelación basada en agentes?
Es una metodología bottom-up que permite modelar y simular computacionalmente fenómenos complejos de distinta índole; ya sean físicos, biológicos, sociales, económicos, etc. con el objetivo de entender y explicar su funcionamiento.

2) ¿Qué es el enfoque bottom-up?
Se considera a los agentes de manera individual con la flexibilidad de establecer características particulares y relaciones con otros agentes y el entorno. Analizamos un sistema compuesto por sus agentes individuales que interactúan entre sí.

3) ¿Cuándo se usa el concepto de autoorganización en sistemas?
Dentro del contexto de los sistemas complejos, un sistema es autoorganizado cuando se producen patrones de manera espontánea sin un plan determinado.

4) ¿Qué es una propiedad emergente?
Los sistemas complejos se componen de múltiples elementos que interactúan entre sí. Estas interacciones pueden producir propiedades emergentes (comportamientos inesperados que surgen de la interacción entre los componentes de una aplicación y su entorno) que no pueden atribuirse a un solo elemento y que los diseñadores del sistema no esperaban.
Es aquella propiedad que se encuentra en un sistema, pero no en sus componentes. La información emergente viene de las interacciones.

Parte 2. Autómata Celular Elemental.

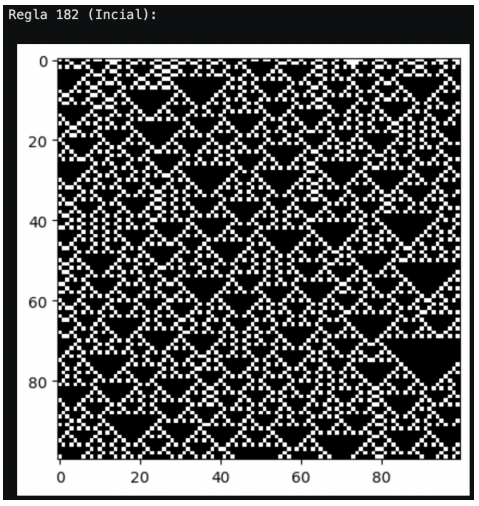
Autómata celular elemental implementado en python.
Ejercicios:

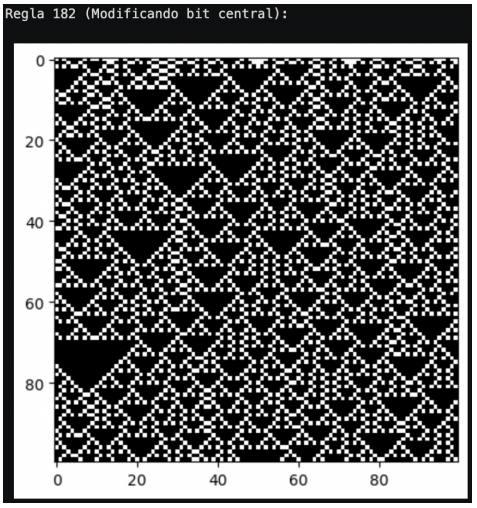
1) La regla 0 para un autómata celular unidimensional es una regla en la que todas las configuraciones posibles de una célula y sus vecinas dan lugar a un estado de salida de 0. Es decir, no importa cuál sea la configuración de tres células vecinas (las combinaciones de 000, 001, 010, 011, 100, 101, 110, 111), el estado de la célula central en el siguiente paso de tiempo siempre será 0.

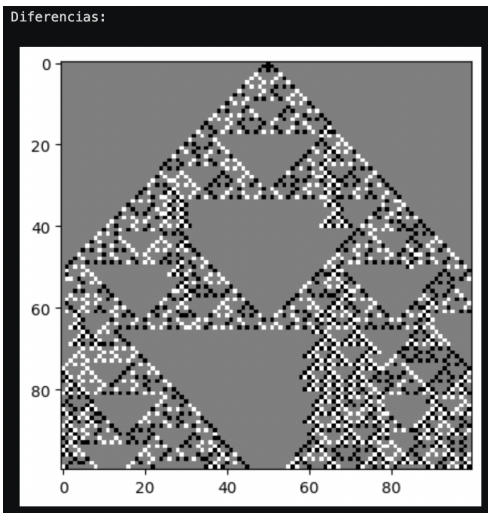
   La regla 0 pertenecería a la Clase 1. Los autómatas de esta clase evolucionan hasta un estado homogéneo. En otras palabras, las configuraciones iniciales tienden a un patrón uniforme cuando se aplica la regla durante un tiempo suficiente. En el caso de la regla 0, este patrón es una línea de ceros. No hay cambios, estructuras o comportamientos más complejos, simplemente todo se convierte en cero.

Haciendo pruebas con diferentes reglas, nosotros pensaríamos que la clase 2 es la más frecuente. Una razón podría ser que en todas las posibles reglas, hay simplemente más formas de alcanzar un estado estable o cíclico que de generar comportamientos complejos o caóticos. Además, estos autómatas suelen ser menos sensibles a las condiciones iniciales, lo que hace que una amplia gama de estados iniciales conduzca a un comportamiento de Clase 2.

2)





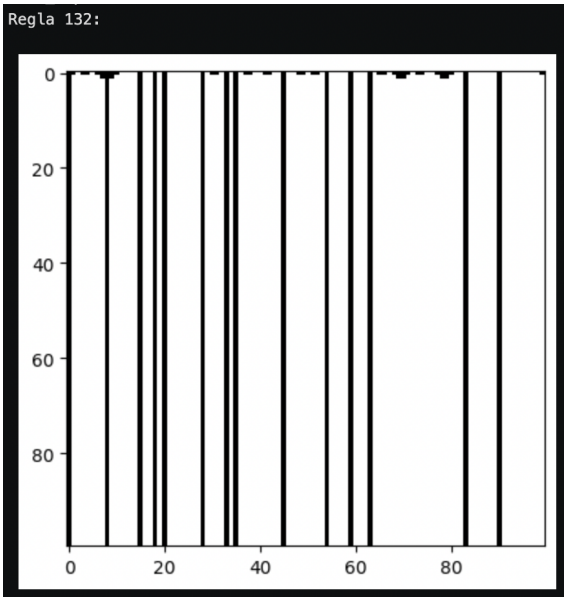


En este caso realizamos la prueba con la regla 182, que pertenece a la clase 3. El primer estado del primer ACE es aleatorio y en el segundo ACE modificamos el bit central del estado inicial. En la tercera imagen vemos las diferencias entre el ACE original y el ACE modificado, el área gris marca posiciones donde los bits no son diferentes, y los bits blancos o negros muestran areas donde los bits son diferenteds. Podemos ver que la pequeña perturbación inicial se va amplificando con el tiempo.

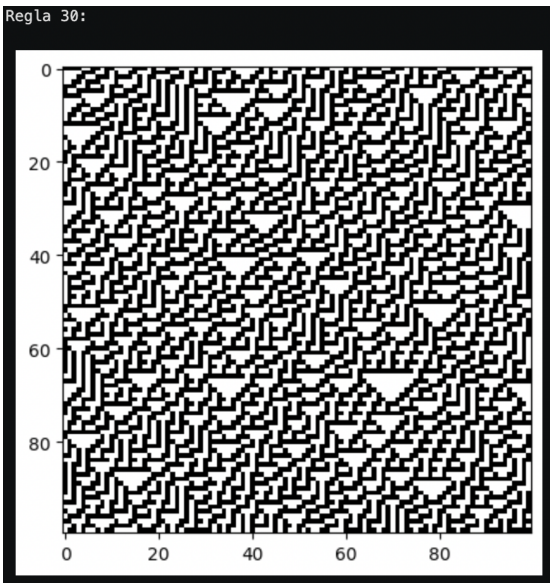
¿La sensibilidad a las condiciones iniciales es una propiedad suficiente para catalogar la dinámica como caótica?

Si una pequeña perturbación inicial causa un gran cambio, eso sugiere que el sistema podría ser caótico, pero no nos da la historia completa. También necesitamos mirar otras propiedades, como si el sistema produce patrones complicados y cambiantes, para estar más seguros de que es caótico.

3) La regla 132 pertenece a la clase 2. Esto se debe a que rápidamente llega a un patrón repetitivo



La regla 30 produce patrones que parecen ser aleatorios y caóticos, produce comportamiento que es complejo.



4) Cuando se comienza con una sola celda negra en el centro, la Regla 22 produce un patrón que se expande a partir de ese punto inicial. La simetría de la condición inicial se refleja en la evolución del sistema: se formará una estructura que es simétrica respecto al punto central. Se produce una especie de triángulo de estructuras que emergen del punto central. La regla pertenece a la clase 2 ya que converge rápidamente a un estado repetitivo o estable.

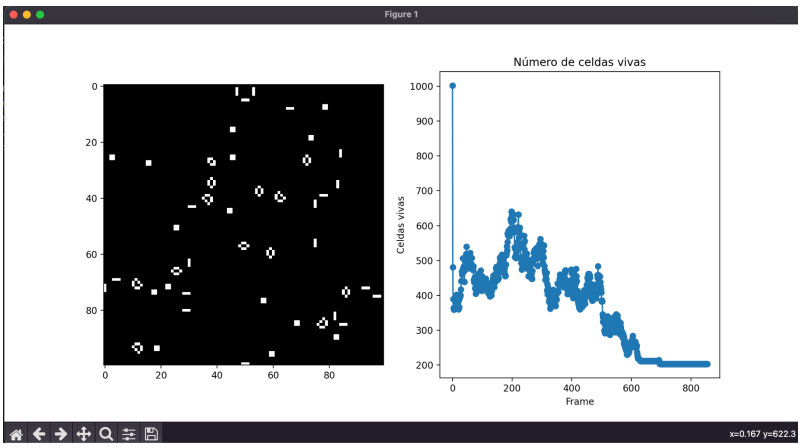
   En cambio cuando se inicia con un estado aleatorio, se muestra una mezcla de patrones que parecerán más irregulares y aleatorios, haciendo parecer que la regla pertenece a la clase 3.

5) Cada grupo local de 3 células tendría 3×3×3=27 posibles configuraciones. Para cada una de estas 27 configuraciones, la célula central podría evolucionar hacia uno de los 3 posibles estados en el siguiente paso de tiempo. Por lo tanto, el número total de posibles reglas sería: 3^27 = 7,625,597,484,987

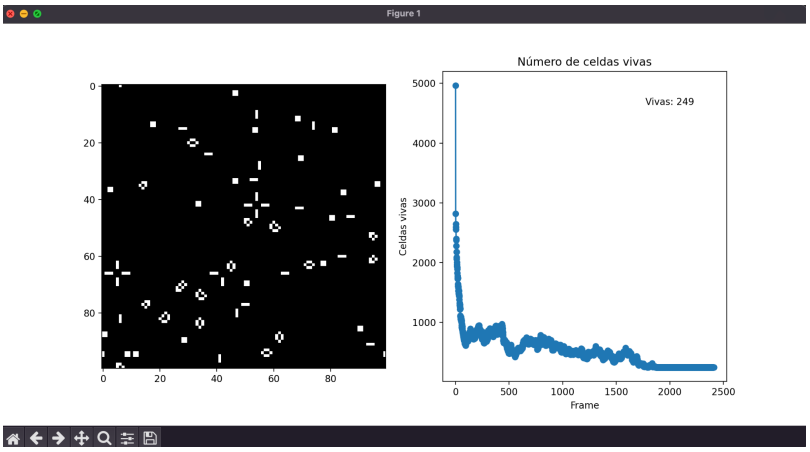
Parte 3. LIFE: EL JUEGO DE LA VIDA.

1)

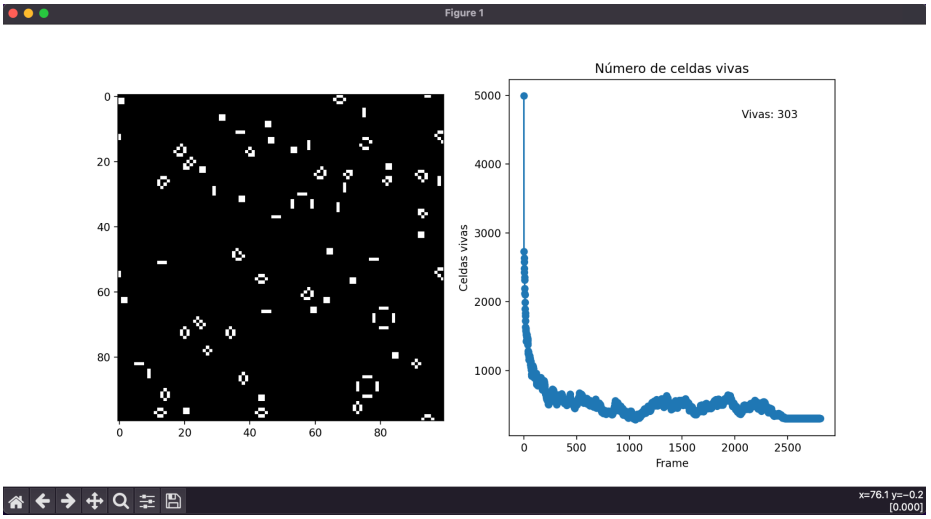
Si iniciamos con un porcentaje de 10% vivas, podemos ver que a los aproximadamente 700 tics, el número de celdas vivas se estabiliza a 200.



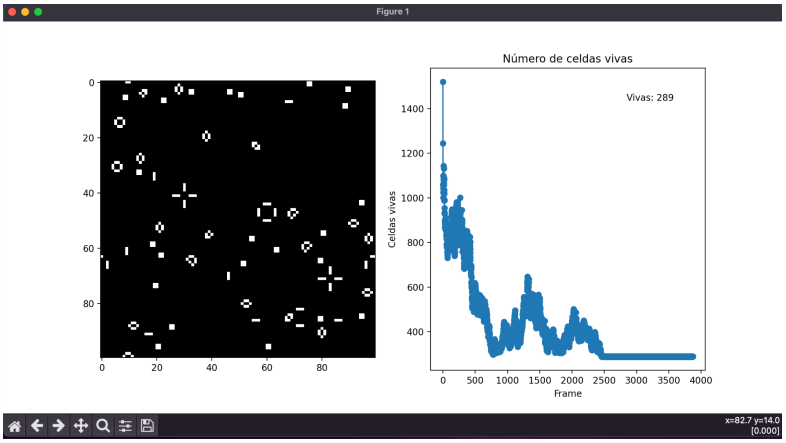
Si iniciamos con un porcentaje de 50% vivas, podemos ver que a los aproximadamente 1700 tics, el número de celdas vivas se estabiliza a 249



Otro intento con un porcentaje de 50% vivas, se estabiliza a los 2500 tics con 303 celdas vivas

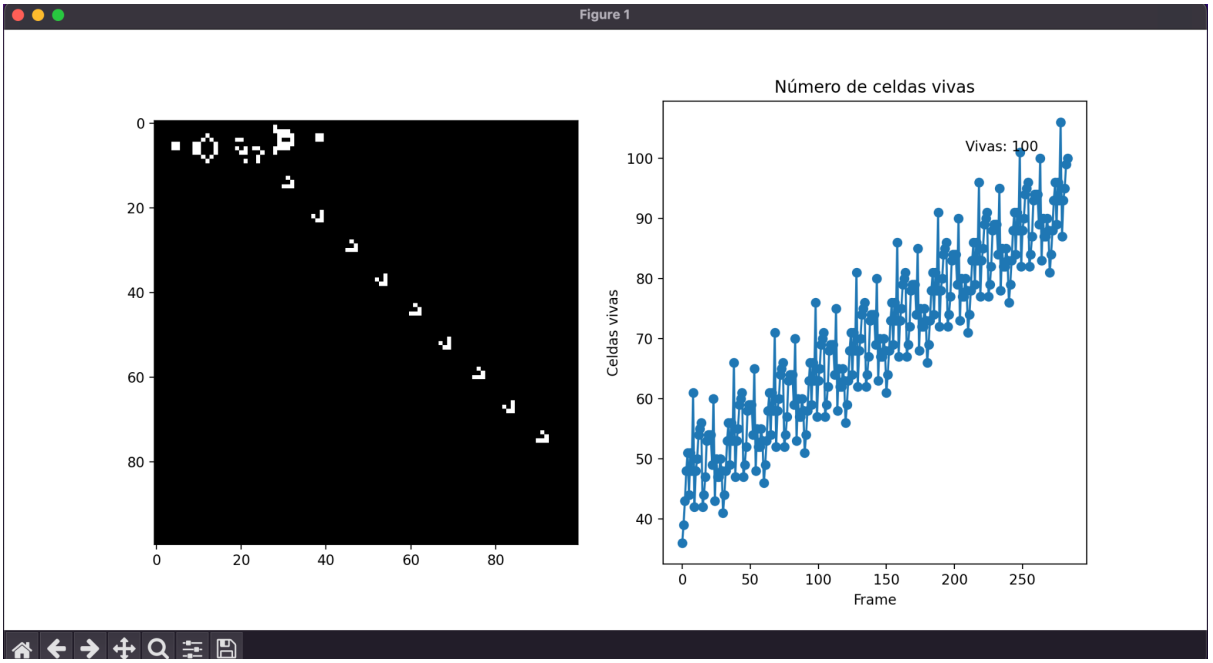


Si iniciamos con un porcentaje de 25% vivas, podemos ver a a los aproximadamente 2500 tics, el número de celdas vivas se estabiliza a 289.



¿Pasará lo mismo para cualquier configuración inicial aleatoria?
Aunque las tendencias generales relacionadas con la densidad inicial podrían mantenerse, el número exacto de tics para la estabilización o el número exacto de celdas vivas en la estabilización no son los mismos para cada ejecución, especialmente con condiciones iniciales aleatorias. El juego de la vida es muy sensible a las condiciones iniciales, y pequeñas variaciones pueden llevar a resultados muy diferentes.

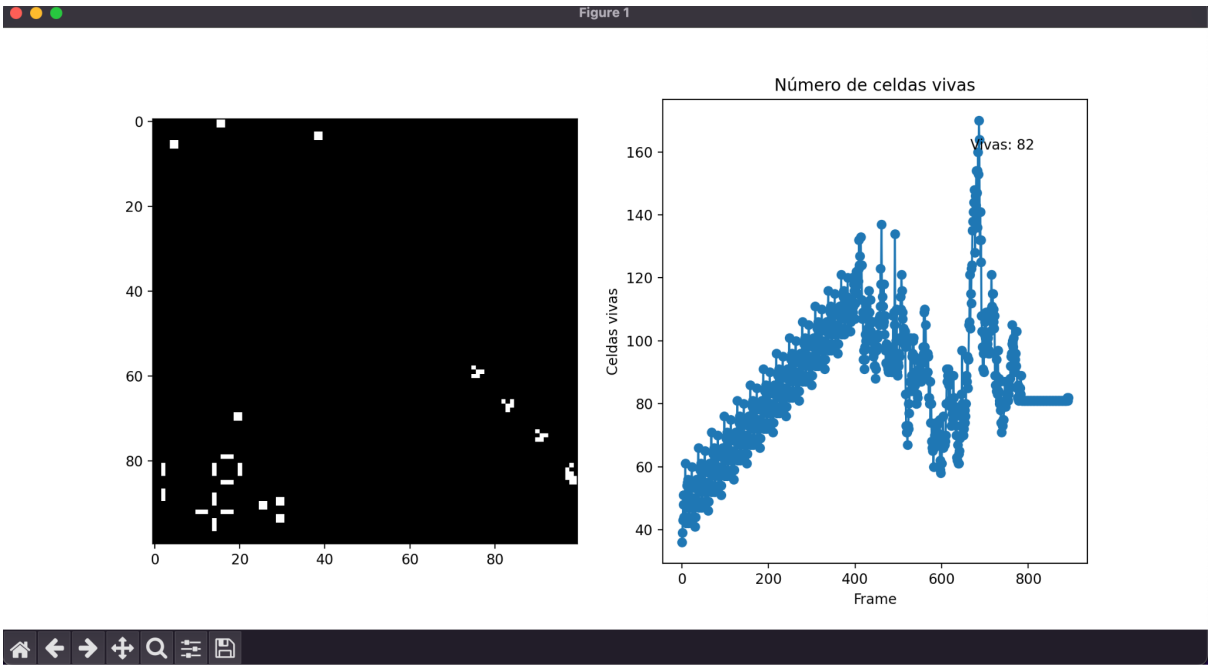
2) Gospers Gun



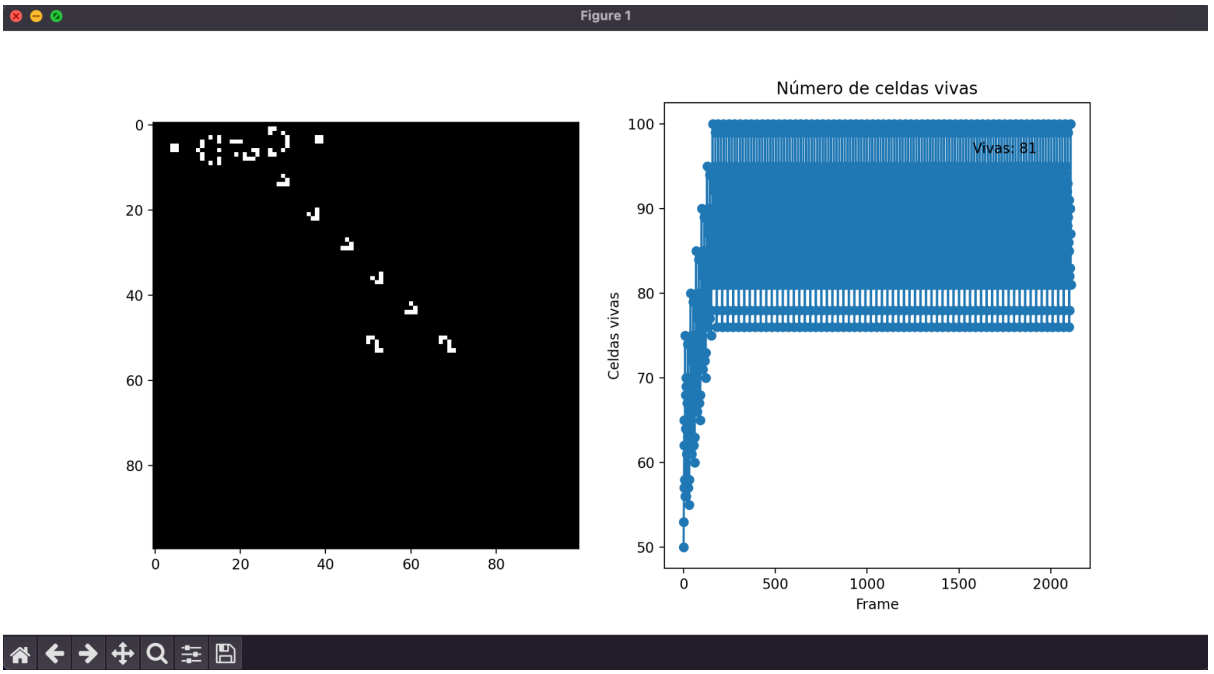
Podemos ver que el gospers gun genera “gliders” o planeadores mientras que la estructura original se mantiene en su lugar. Dispara un nuevo planeador cada 30 iteraciones (o "ticks"). Por lo tanto, aunque la configuración cambia con el tiempo, la dinámica es periódica en el sentido de que sigue un patrón cíclico. Después de cierto tiempo, nuestro Gospers gun se deshace, no estamos seguros si es algo que debería suceder o es un error en nuestra configuración. Podría ser que debido al tamaño de nuestro tablero, ocurra cierta

configuración que deshaga el patrón, pero en un tablero infinito este patrón debería repetirse para siempre.

Gosper's Gun deshecho 😕:



3) Comedor "Eater"



Implementamos un "eater" que come las "balas" del guspers gun. Aquí notamos que ahora el patrón sí se repite de manera periódica de manera "infinita".

4) **El juego de la vida es reversible?**
   El Juego de la Vida de Conway no es reversible porque no es posible determinar unívocamente un estado anterior a partir de un estado dado. La actualización de cada celda sobrescribe información sobre su estado anterior, y hay múltiples configuraciones iniciales que pueden llevar al mismo estado futuro, lo que hace que la reversión del proceso sea ambigua o imposible.