Implementación del Juego de la vida de Conway en PowerDEVS

Lucio Mansilla

2 de agosto de 2023

1. Introducción / Problema

El Juego de la Vida de Conway, comúnmente conocido como "Juego de la Vida", es un autómata celular que fue propuesto por el matemático británico John Horton Conway en 1970. Los autómatas celulares son modelos matemáticos para sistemas dinámicos que evolucionan en pasos de tiempo discretos(generaciones). A pesar de su simplicidad aparente, tienen la capacidad de simular sistemas complejos y mostrar comportamientos emergentes, siendo utilizados en diversos contextos, desde la física hasta la teoría de computación.

En el Juego de la Vida, cada celda en una cuadrícula bidimensional puede estar en uno de dos estados: "viva.º "muerta". Las celdas interactúan con sus ocho vecinos adyacentes en horizontal, vertical y diagonal, transicionando entre los estados de vida y muerte de acuerdo a las siguientes reglas de evolución:

- Cualquier celda viva con dos o tres vecinos vivos sobrevive para la siguiente generación.
- Cualquier celda viva con menos de dos vecinos vivos muere por soledad para la siguiente generación.
- Cualquier celda viva con más de tres vecinos vivos muere por superpoblación para la siguiente generación.
- Cualquier celda muerta con exactamente tres vecinos vivos nace para la siguiente generación.

Aunque estas son las reglas originales, existen muchas variantes del Juego de la Vida con diferentes reglas. Además, es posible observar la aparición de diversos patrones, algunos estáticos, otros oscilan entre varios estados y otros se desplazan por el tablero. Estos patrones serán el objeto de estudio en las secciones posteriores de este informe.

El propósito de este proyecto es explorar la dinámica del juego de una manera visual e interactiva, para ello se implementará el mismo en PowerDEVS, una herramienta de simulación de eventos discretos basada en la teoría DEVS (Discrete Event System Specification).

2. Especificación DEVS de una celda

El DEVS que representa únicamente a una célula se define como sigue:

$$C = \langle X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta \rangle$$

donde

- $X = GameState$

- $Y = \mathbb{N} \times \{0, 1\}$
- $S = \mathbb{N} \times \{0, 1\} \times \mathbb{R}_0^+$

  El estado es una tupla (cid, ls, σ) donde

  - $cid \in \mathbb{N}$ es el identificador de la célula.
  - $ls \in \{0, 1\}$ es el estado de la célula (1 = viva, 0 = muerta)
  - $\sigma \in \mathbb{R}_0^+$ es el tiempo restante para realizar una próxima salida.
- $\delta_{\text{int}}((cid, ls, \sigma)) = (cid, ls, \infty)$
- $\delta_{\text{ext}}((cid, ls, \sigma), e, (x, p)) = \begin{cases} (cid, 0, 1) & ls == 1 \wedge \notin SR \\ (cid, 1, 1) & ls == 0 \wedge \in BR \\ (cid, ls, 1) & \text{otherwise} \end{cases}$

  donde $alives = \text{countAlives}(x.rows, x.cols, cid, x.board)$
- $\lambda((cid, ls, \sigma)) = (cid, ls)$
- $ta((cid, ls, \sigma)) = \sigma$
- **countAlives**: La función **countAlives** tiene como objetivo calcular el número de células vecinas que están vivas para una célula dada. Esta función recibe cuatro parámetros: el número de filas (rows) y columnas (cols) del tablero, el identificador de la célula (cell_id) y el estado actual del tablero (board). La función opera de la siguiente manera:
- Primero, determina la ubicación (fila y columna) de la célula dada en el tablero usando su identificador (cell_id) y las dimensiones del tablero.
- Posteriormente, recorre todas las células vecinas de la célula dada. Esto se realiza iterando a través de todas las posibles combinaciones de desplazamientos en filas y columnas en el rango de -1 a 1.
- Ignora la propia célula durante la iteración, es decir, el caso cuando tanto el desplazamiento de la fila como de la columna es cero.
- Para cada célula vecina, primero verifica si está dentro de los límites del tablero. Si la célula vecina está fuera de los límites del tablero, pasa a la siguiente iteración.
- Si la célula vecina está dentro de los límites, verifica si está viva, es decir, si su estado en el tablero es 1. Si es así, incrementa un contador de células vivas.
- Finalmente, después de haber iterado a través de todas las células vecinas, la función devuelve el contador de células vivas.

3. Implementación en PowerDEVS

4. Patrones

El Juego de la Vida, concebido por el matemático británico John Horton Conway, es famoso por la vasta variedad de patrones que emergen de sus simples reglas, en particular, las reglas 23/3. Los patrones se definen como configuraciones de células que se repiten tras un número determinado de generaciones, también conocido como su "periodo". Los patrones pueden ser estáticos, oscilantes o móviles, dependiendo de cómo cambian a lo largo del tiempo. En las secciones siguientes se presentarán ejemplos de estos patrones bajo las reglas 23/3.

4.1. Patrones estáticos

Los patrones estáticos, también conocidos como "still lifes", son configuraciones de células que no cambian de una generación a la siguiente, es decir, su periodo es 1. A continuación, se presentan ejemplos de estos patrones:

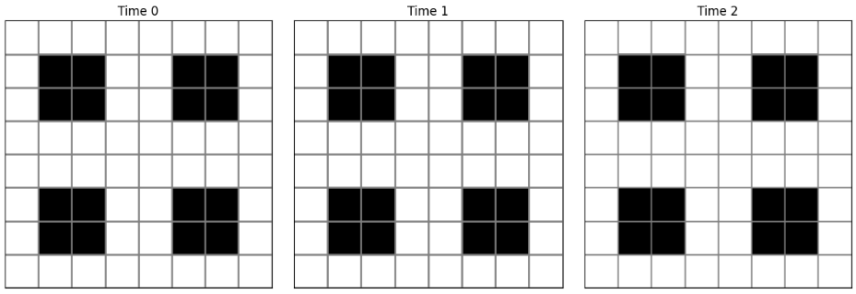

Figura 1: Patrón bloque: un patrón estático compuesto por un cuadrado de 2x2 células. Aparece frecuentemente debido a su simplicidad.

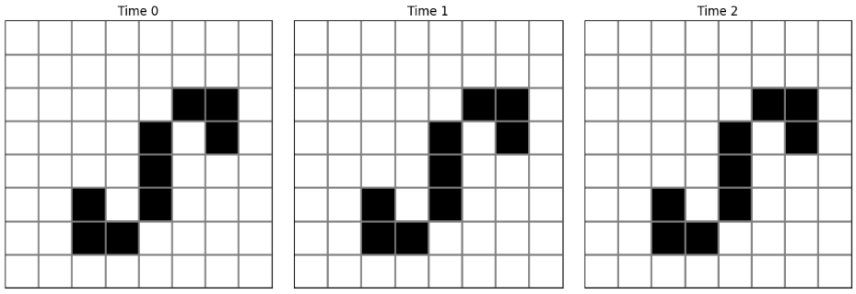

Figura 2: Patrón integral: este patrón estático se asemeja a la forma de un signo integral.

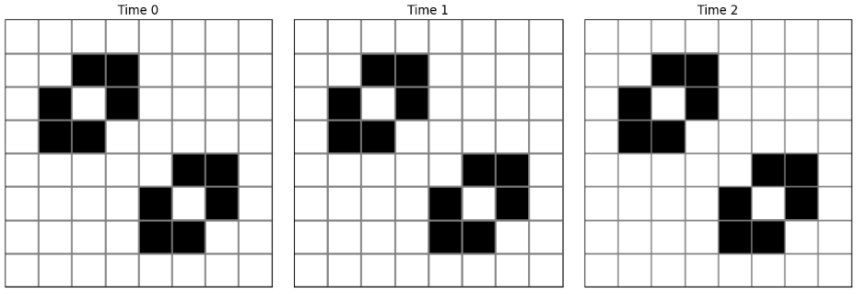

Figura 3: Patrón ship: este patrón estático se asemeja a la forma de una nave espacial.

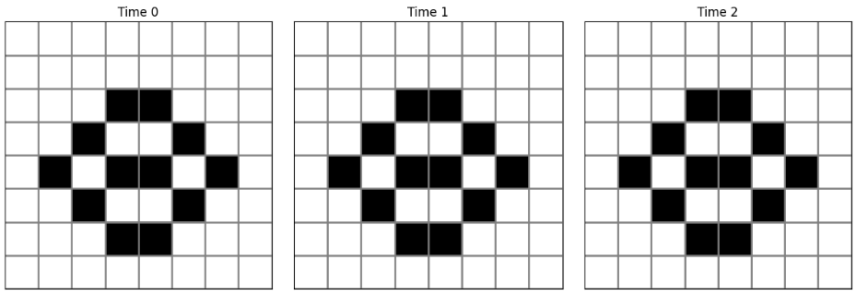

Figura 4: Patrón honeycomb: este patrón estático se asemeja a la estructura de un panal de abejas.

4.2. Osciladores

Los osciladores son patrones que vuelven a su configuración inicial después de un número fijo de generaciones, llamado su período. A diferencia de los patrones estáticos, los osciladores cambian su apariencia durante su ciclo de vida, pero eventualmente vuelven a su estado original. Un ejemplo famoso de oscilador es el "blinker", que oscila entre una orientación horizontal y vertical cada generación.

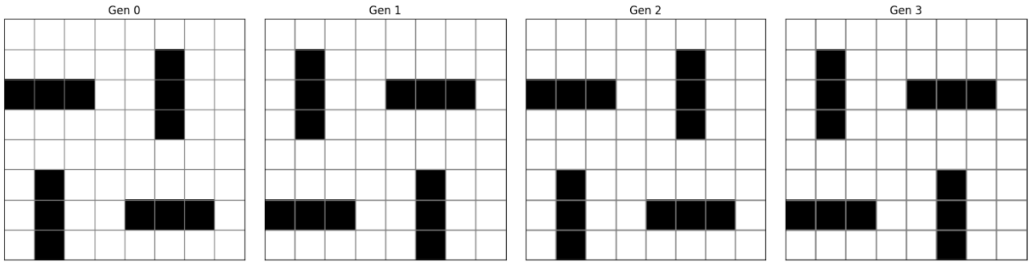

Figura 5: Patrón blinker: un oscilador de periodo 2. Oscila entre una orientación horizontal y vertical cada generación. Notar que en este caso hay 4 blinkers en el tablero.

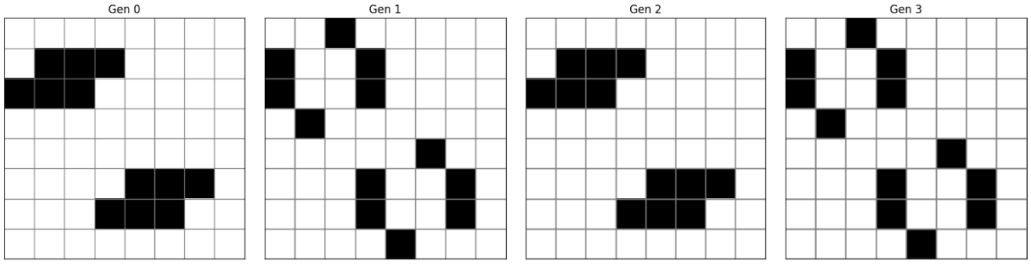

Figura 6: Patrón toad:

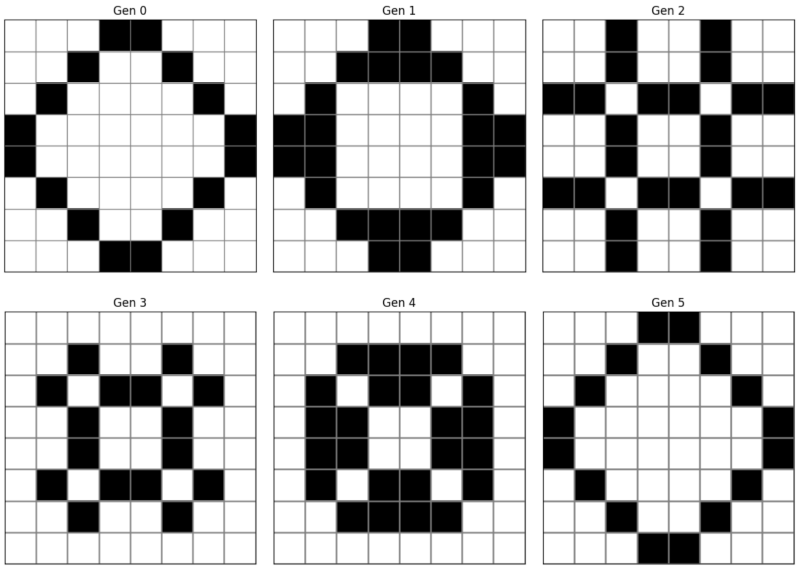

Figura 7: Patrón octagon:

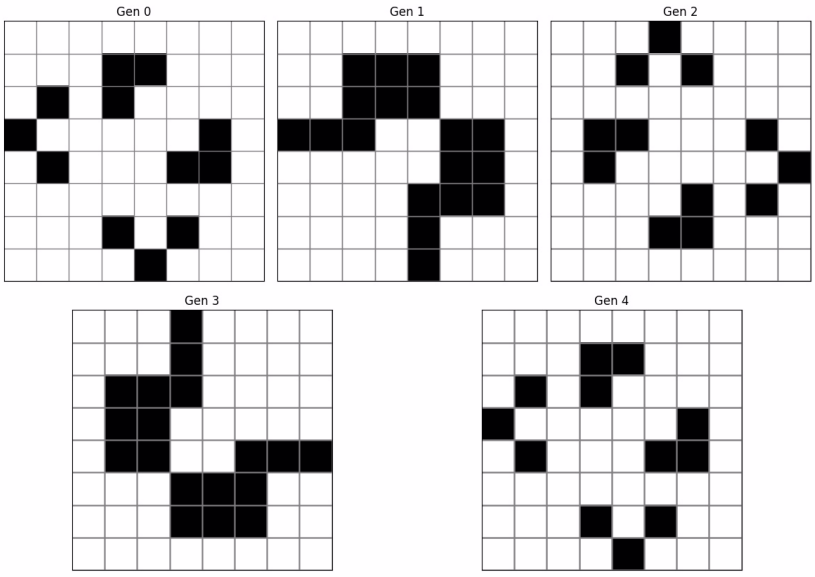

Figura 8: Patrón octagon:

4.3. Naves espaciales

Las naves espaciales son patrones que se trasladan a través de la cuadrícula mientras oscilan. El ejemplo más conocido es la "nave ligera"(o "glider.[en] inglés), que se desplaza diagonalmente a través de la cuadrícula mientras oscila entre cuatro configuraciones diferentes.

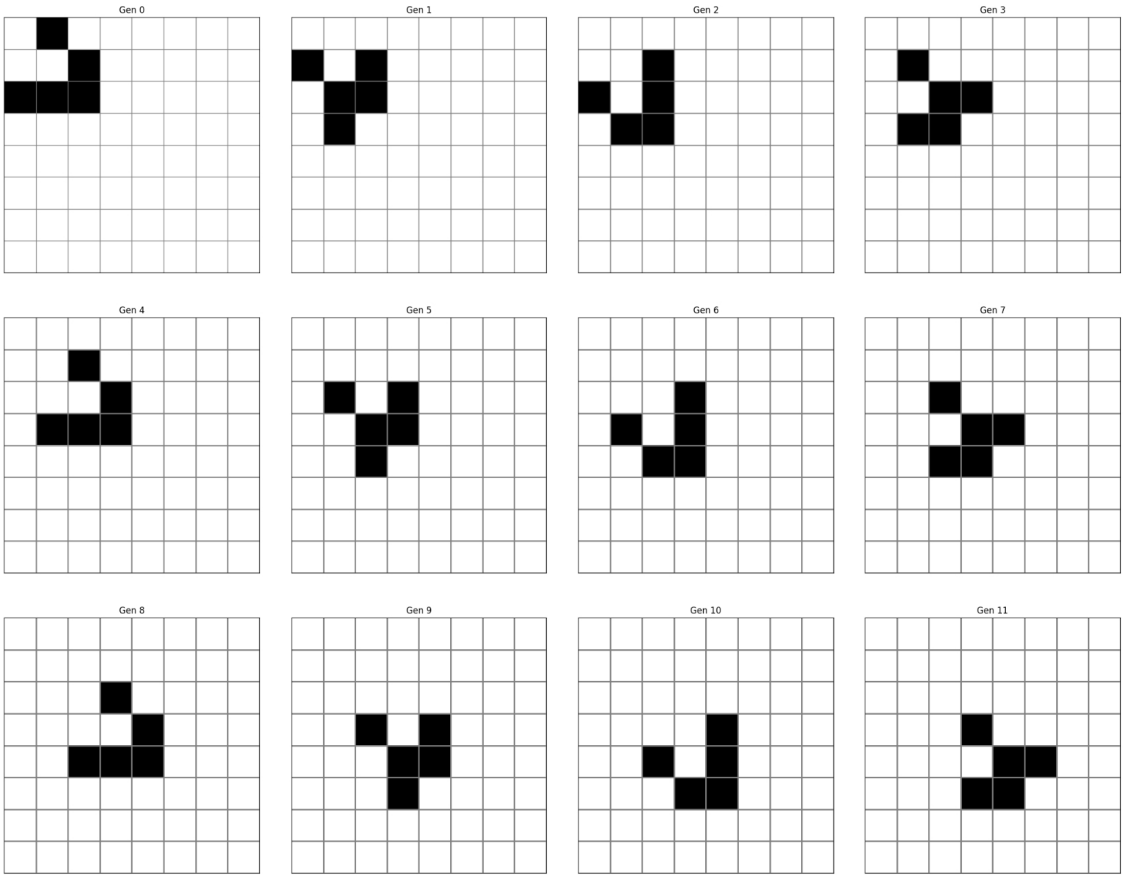

Figura 9: Patrón glider:

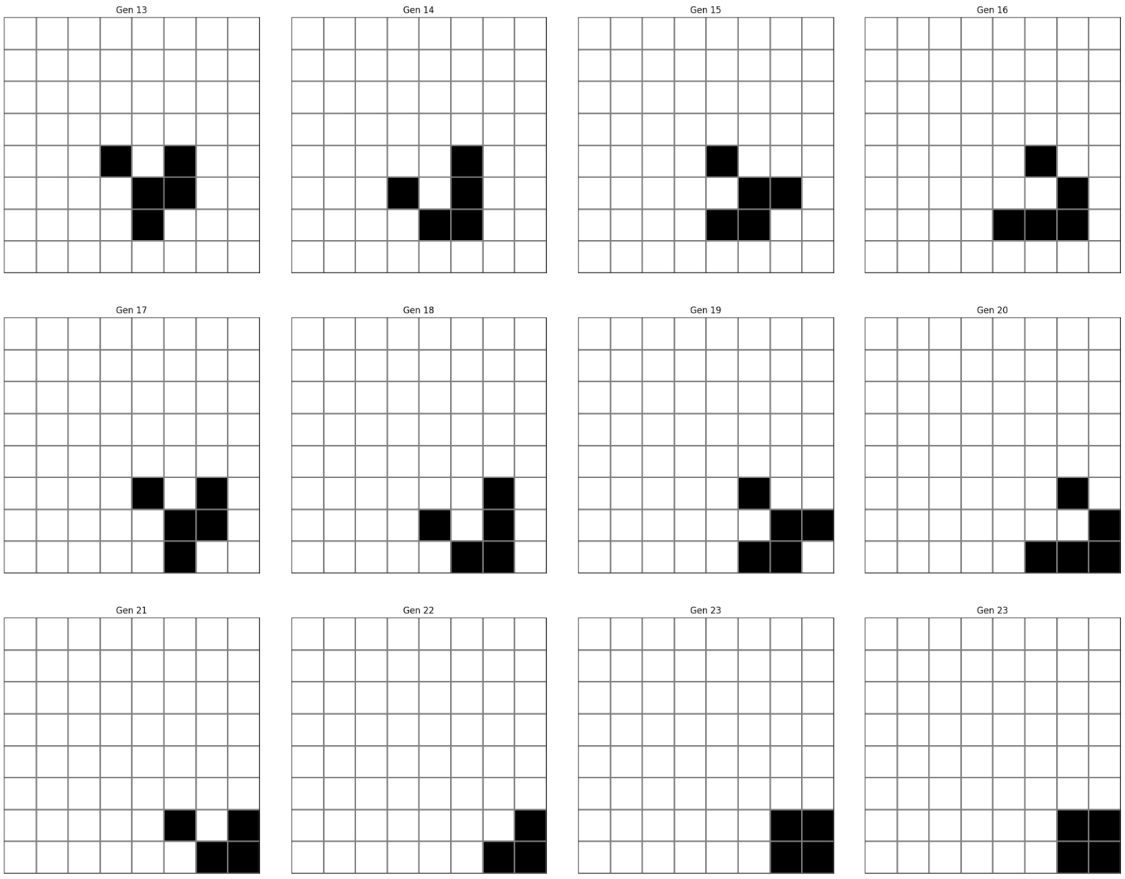

Figura 10: Patrón glider continuación:

4.4. Methuselahs

Los Methuselahs son patrones que evolucionan durante un número grande de generaciones antes de estabilizarse en un patrón estático, un oscilador o una nave espacial. El ejemplo más conocido es el .ªcorn", que evoluciona durante 5206 generaciones antes de estabilizarse en 633 células vivas, incluyendo 11 osciladores, 2 naves espaciales y 1 patrón estático.

4.5. Patrones Propios

Sección dedicada a patrones que encontramos por accidente/pruebas.

4.6. Patrones complejos

Además de estos patrones simples, también existen patrones más complejos en el Juego de la Vida. Algunos de estos pueden "disparar"naves espaciales, otros pueden çonstruir"patrones adicionales, y otros pueden incluso comportarse como máquinas de Turing universales, lo que significa que pueden computar cualquier función computable.

Estos patrones ilustran la increíble diversidad de comportamientos que pueden surgir del Juego de la Vida. En las siguientes secciones, veremos cómo estos patrones pueden ser generados e investigados usando PowerDEVS.

5. Conclusiones