Implementación del Juego de la vida de Conway's en PowerDEVS

Lucio Mansilla - Brenda Dichiara

4 de agosto de 2023

1. Introducción / Problema

El Juego de la Vida de Conway's, 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 que compone la cuadricula bidimensional puede estar en uno de dos estados: *viva* o *muerta*. Las celdas a lo sumo 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 en su versión original:

- 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/aislamiento para la siguiente generación.
- Cualquier celda viva con más de tres vecinos vivos muere por sobrepoblación para la siguiente generación.
- Cualquier celda muerta con exactamente tres vecinos vivos nace para la siguiente generación.

Estas reglas originales son comúnmente denotadas como 23/3, representando las condiciones de supervivencia y nacimiento, respectivamente. Es decir, una celda viva sobrevive si tiene 2 o 3 vecinos vivos y una celda muerta nace si tiene exactamente 3 vecinos vivos. Se han desarrollado una gran cantidad de variantes. En este sistema de reglas, la denominación "S/B"(Survive/Birth) representa las condiciones para la supervivencia y el nacimiento de las células. "S"son los números de vecinos que permiten a una celda viva sobrevivir, y "B"son los números de vecinos que permiten a una celda muerta nacer. Algunas variantes que podemos destacar:

- HighLife: 23/36. Similar al Juego de la Vida original, pero con la adición de que las células muertas con 6 vecinos vivos también nacen. Este juego es famoso por su patrón replicator", el primero que se descubrió y que puede replicarse a sí mismo, se abordará en más detalle en la sección 4.6.
- Seeds: 0/2. En esta variante, las células vivas no sobreviven y las células muertas nacen si tienen exactamente 2 vecinos vivos. Es conocido por su crecimiento muy rápido y la creación de estructuras complejas a partir de pequeñas semillas.

Es importante señalar que la mayoría de las 2^{18} posibles reglas del espacio producen universos que son o bien demasiado caóticos, o bien demasiado desolados para ser de interés. Sin embargo, 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 juego en PowerDEVS, una herramienta de simulación de eventos discretos basada en la teoría DEVS (Discrete Event System Specification).

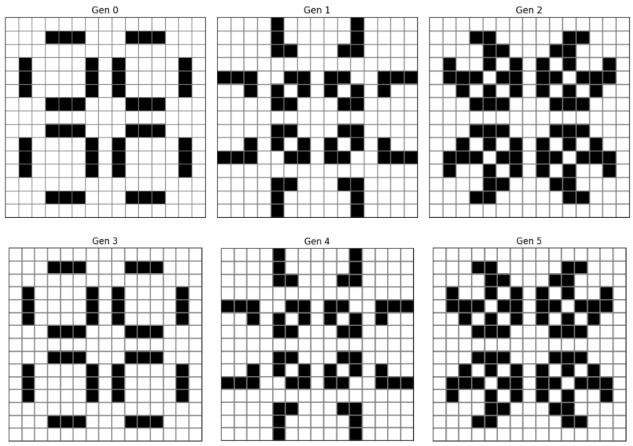

Figura 1: Patrón Pulsar: oscilador de periodo 3.

2. Especificación DEVS de una celda

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

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

donde

- $X = GameState$

  La estructura de $GameState$ se define formalmente como:

  - **board**: Un conjunto que representa el estado de cada célula en el tablero del juego. Cada celda en el tablero puede tener un valor de 0 (muerta) o 1 (viva).
  - **rows**: El número total de filas en el tablero.
  - **cols**: El número total de columnas en el tablero.
  - **SR**: Representa las reglas de supervivencia.
  - **BR**: Representa las reglas de nacimiento.

- $Y = \mathbb{N} \times \{0, 1\}$

  La salida esta compuesta por la tupla (cid, ls)

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

  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.

- $\delta_{\text{int}}((cid, ls, \sigma)) = (cid, ls, \infty)$

  Luego de enviar su estado, la célula espera a que se le envíe una nueva entrada.

- $\delta_{\text{ext}}((cid, ls, \sigma), e, (x, p)) = \begin{cases} (cid, 0, 1) & ls = 1 \ \wedge \ alives \ \notin x.SR \\ (cid, 1, 1) & ls = 0 \ \wedge \ alives \in x.BR \\ (cid, ls, 1) & \text{otherwise} \end{cases}$

  donde $alives = \text{countAlives}(x.rows, x.cols, cid, x.board)$

- $\lambda((cid, ls, \sigma)) = (cid, ls)$
- $ta((cid, ls, \sigma)) = \sigma$

Donde $x.SR$ y $x.BR$ representan las reglas de supervivencia y nacimiento respectivamente, las cuales se explican en detalle en la Sección 1: "Introducción / Problema".

- **countAlives**: Esta función es responsable de determinar la cantidad de células vecinas vivas para una célula específica. Se le proporcionan cuatro parámetros claves: el número de filas (rows) y columnas (cols) del tablero, el identificador único de la célula (cell_id) y el estado actual del tablero (board). La función realiza tres pasos principales:
  - **Identificación**: Localiza la posición exacta de la célula dada en el tablero utilizando su identificador y las dimensiones del tablero.
  - **Exploración y Conteo**: Examina únicamente las células vecinas, excluyendose a sí misma, y realiza un conteo de cuantas vecinas estan vivas.
  - **Resultado**: Retorna la cantidad total de células vecinas vivas.

2.1. Mediator

Como podemos apreciar, la lógica de cada célula en DEVS es relativamente sencilla. Esto se debe, en gran medida, a la presencia de un patrón de diseño muy popular en la programación conocido como *Mediator*. El Mediator permite desacoplar la lógica de las células, gestionar el tiempo de los eventos y distribuir la información del tablero. Es un patrón de diseño de comportamiento que se utiliza para reducir las comunicaciones complejas y las dependencias entre diferentes entidades.

En el contexto de DEVS, el *Mediator* se encarga de transmitir el estado inicial, así como cualquier cambio en el tablero a las células individuales. A su vez, las células informan al *Mediator* sobre cualquier cambio de estado. Este proceso permite que la información se distribuya de manera independiente entre todas las células.

Esta estrategia trae consigo una serie de ventajas significativas. En primer lugar, se simplifica el diseño de las células al no tener que lidiar con la lógica adicional para manejar los tiempos sincronizados de los eventos discretos (Por ejemplo, tener que primero actualizar y luego informar o viceversa). En segundo lugar, al eliminar la necesidad de que cada célula esté directamente conectada con todas las demás, se obtiene un sistema más flexible, independiente y escalable, puesto que en un futuro solo basta con añadir una nueva célula y conectarla al *Mediator* para que esta pueda interactuar con el resto del sistema siempre que se obtenga un tablero de n filas y m columnas.

La especificación DEVS del *Mediator* se puede describir como:

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

- $X = \mathbb{N} \times \{0, 1\}$

  La entrada es la tupla$(cell_id, life_state)$ proveniente de una célula.

- $Y = GameState$

  La salida es el estado del juego junto con sus dimensiones y reglas.

- $S = GameState \times \mathbb{R}_0^+$

  definiendo game $\in GameState$ entonces se tiene:

- $\delta_{\text{int}}(game, \sigma) = (game, \infty)$

  El mediator luego de enviar la información a lás células espera a que las mismas se comuniquen con él.

- $\delta_{\text{ext}}(game, \sigma, (cell\_id, life\_state)) = (game.board[x.cell\_id] \leftarrow x.life\_state, 1)$

  En el tablero de juego, se actualiza el nuevo estado de la célula con identificador $cell\_id$ y se establece el tiempo de la próxima salida en 1.

- $\lambda(game) = game$

- $ta(game, \sigma) = \sigma$

3. Implementación en PowerDEVS

En PowerDEVS, el juego de la vida se implementa utilizando dos componentes claves: una entidad denominada 'célula' y 'mediator'. Ambos son modelos atómicos y sus relaciones e interacciones definen el estado y el comportamiento del juego.

La célula se implementa como un modelo atómico simple con una única entrada y salida. Cada célula es una entidad única que cuenta con un identificador unívoco que la representa en el tablero. Por otro lado, el Mediator es un modelo que juega un papel crucial para coordinar y gestionar las interacciones entre las mismas.

El Mediator, también implementado como un modelo atómico, desempeña un papel central en la coordinación y el control de las interacciones entre las células. Este componente no solo gestiona las relaciones entre ellas, sino que también es responsable de inicializarlas. Recibe como parámetro el nombre del archivo que contiene el estado inicial del tablero asi como las reglas de supervivencia y nacimiento (1er parámetro).

Además una de las funcionalidades importantes del Mediator es que permite definir el nombre de archivo de salida (2do parámetro), en el mismo se registra el estado del tablero del juego en cada generación, facilitando su exportación y visualización en formatos como gif o png. En la Figura 2 a la izquierda se muestra la representación de una célula como un modelo atómico, y a la derecha, el Mediator.

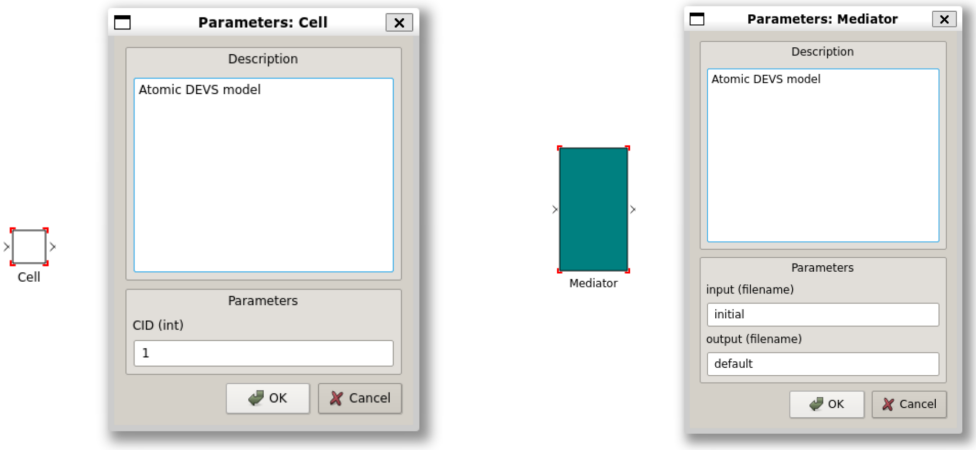

Figura 2: El Mediator y la Célula como modelos atómicos en PowerDEVS.

El trabajo conjunto de estas dos entidades da lugar a la dinámica del juego. A través del Mediator, el juego inicializa y coordina el estado de las células, gestionando su comportamiento a lo largo de cada generación. En la siguiente imagen, se proporcionará un ejemplo de un tablero de 3x3 inicializado por un Mediator, ilustrando cómo se interconectan múltiples células y cómo este permite que cada célula sea independiente de las demás, es decir, no se comunican directamente entre ellas.

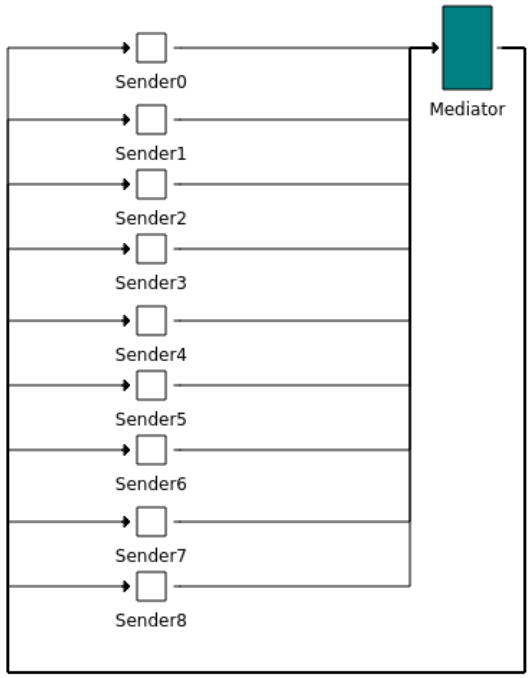

Figura 3: Tablero de ejemplo 3x3.

También se incorporó al proyecto una librería, en la misma se pueden encontrar los modelos atómicos anteriormente mencionados en las figuras 2 y 3, así como también varios modelos de tableros de ejemplo, como se puede observar en la figura 4.

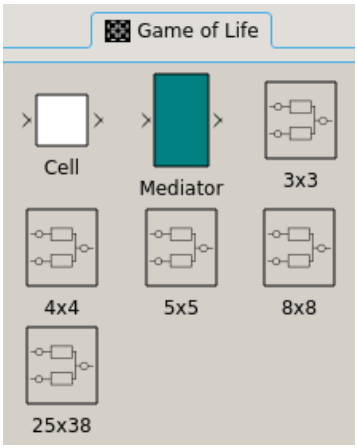

Figura 4: Librería

3.1. Optimización

La optimización implementada en la especificación DEVS de la célula se basa en minimizar las comunicaciones innecesarias con el Mediator. Más específicamente, evita enviar actualizaciones de estado al Mediator cuando el estado de una célula permanece sin cambios. Esto tiene un impacto significativo en la eficiencia del sistema, especialmente para tableros de juego grandes y simulaciones que se ejecutan durante un número considerable de generaciones.

La lógica detrás de esta optimización es que si el estado de una célula permanece igual (es decir, permanece viva o muerta), no hay necesidad de informar al Mediator sobre este evento nulo. Ya que el mismo posee el estado actual del tablero de juego, y un estado de célula inalterado no aporta ninguna información nueva. Por lo tanto, al omitir estas comunicaciones innecesarias, podemos reducir la carga de computación y comunicación.

En una simulación con un tablero de 8x8 y un tiempo de ejecución de $t = 10{,}000$, la implementación

de la misma redujo el tiempo de ejecución de 754 ms a 2 ms.

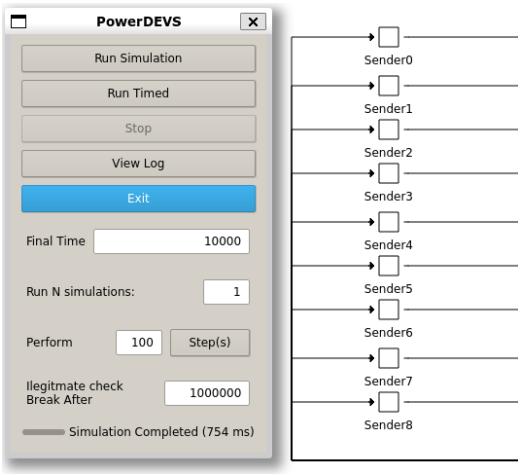

Figura 5: Simulación sin optimización.

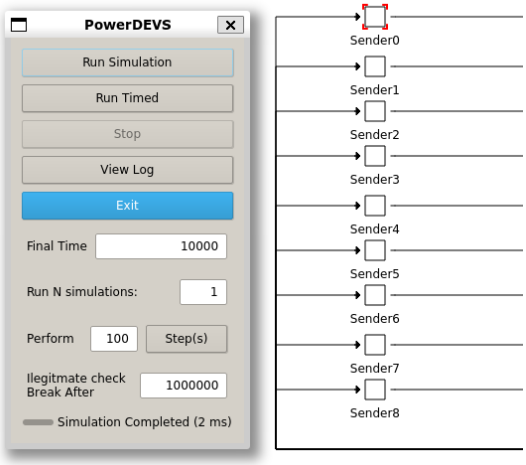

Figura 6: Simulación con optimización.

Notablemente, es especialmente eficaz en patrones donde ciertas áreas del tablero permanecen estáticas durante largos períodos de tiempo. Por ejemplo, en simulaciones con el patrón "glider", se observaron mejoras similares.

3.2. Automatización y Personalización de Tableros

Entendemos que la creación de tableros personalizados en PowerDEVS puede ser un proceso laborioso, especialmente para tableros de grandes dimensiones. Para facilitar este proceso, hemos desarrollado una herramienta de ayuda basada en una interfaz gráfica de usuario (GUI) implementada en Python.

Esta herramienta, que hemos llamado 'Settings GoL', permite generar tableros de dimensiones NxM en formato PDM, los cuales pueden ser utilizados directamente en PowerDEVS. El proceso es sencillo e intuitivo, y reduce significativamente el tiempo y el esfuerzo requeridos para la configuración y configuración de conexiones de tableros personalizados.

En la Figura 7, se muestra una vista previa de la interfaz de Settings GoL y de algunas de sus funcionalidades.

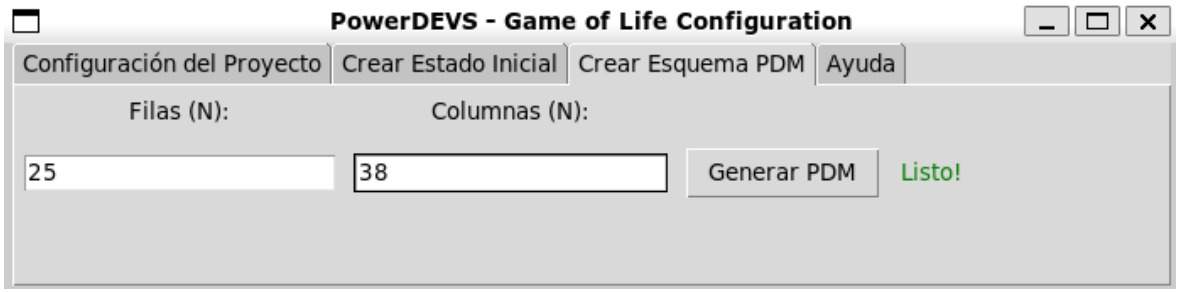

Figura 7: Vista previa de Settings GoL, pestaña generador de tableros PDM.

Para un entendimiento completo de cómo utilizar esta herramienta y sus otras funcionalidades se ha elaborado una guía detallada de uso que se encuentra en el anexo de este informe.

4. Patrones

El Juego de la Vida, es famoso por la vasta variedad de patrones que emergen de sus simples reglas, en particular, las originales 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 mismos 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 y en tableros de 8x8. En la sección 4.5 se explorarán patrones más complejos que requieren tableros de mayor tamaño para su visualización y/u otras reglas.

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:

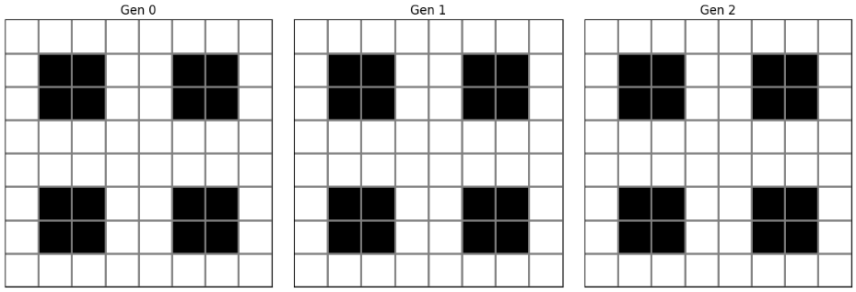

Figura 8: Patrón bloque: un patrón estático compuesto por un cuadrado de 2x2 células. Aparece frecuentemente debido a su simplicidad.

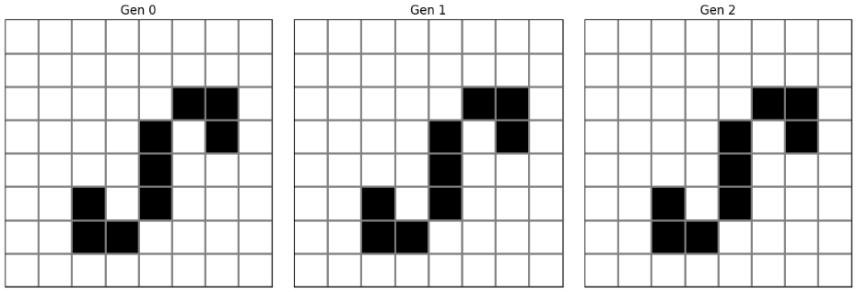

Figura 9: Patrón integral: este patrón estático se asemeja a la forma de un signo integral.

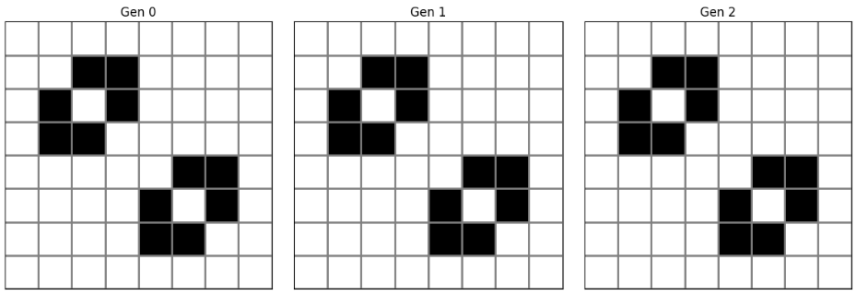

Figura 10: Patrón ship: este patrón estático se asemeja a la forma de un barco

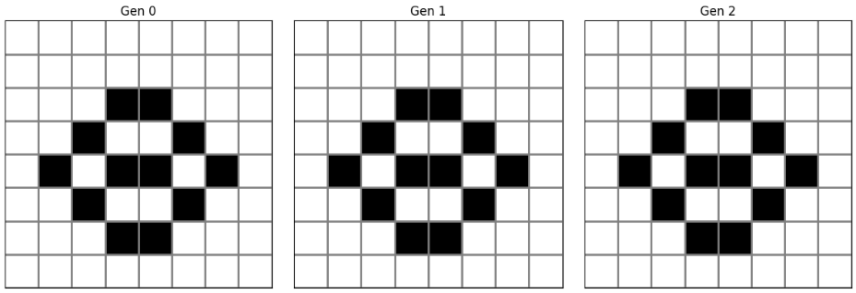

Figura 11: Patrón honeycomb: este patrón estático se asemeja a la estructura de un panal de abejas.

4.2. Osciladores

Los osciladores son patrones que vuelven a su configuración inicial después de un número fijo de generaciones, conocido como 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.

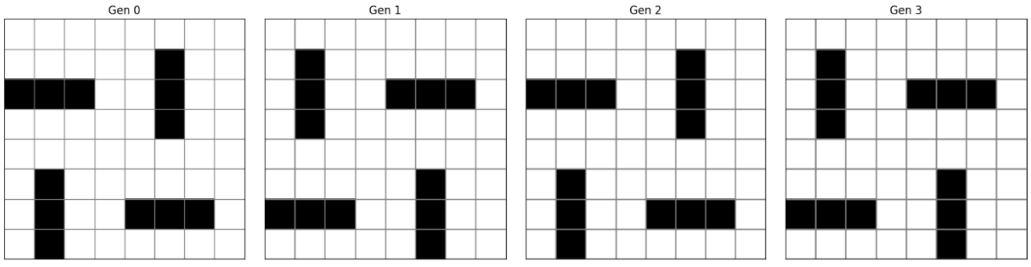

Figura 12: Patrón blinker: un oscilador de periodo 2. Oscila entre una orientación horizontal y vertical cada generación.

Notar que en este caso hay 4 blinkers en el tablero.

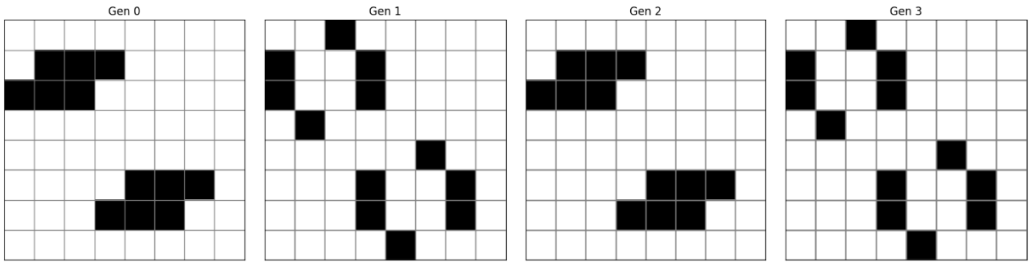

Figura 13: Patrón toad: un oscilador de periodo 2 que oscila entre dos configuraciones.

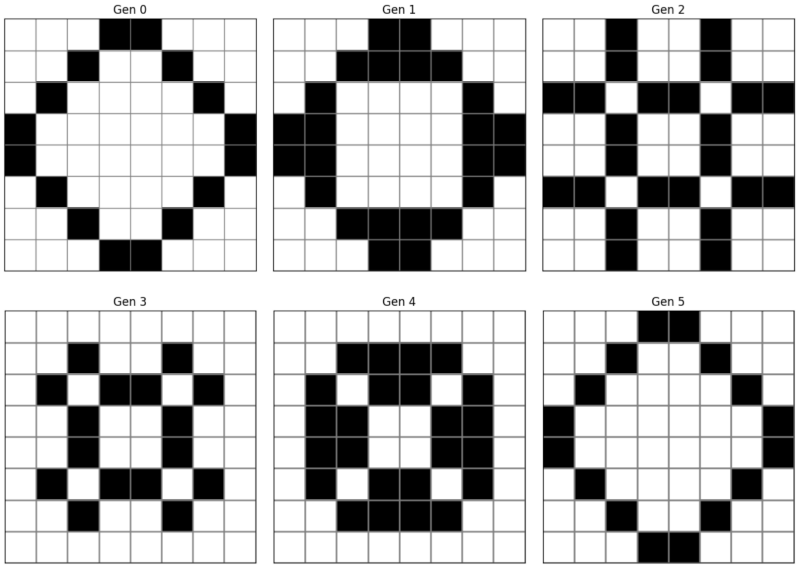

Figura 14: Patrón octagon: un oscilador de periodo 5 que se desplaza en un ciclo de rotación.

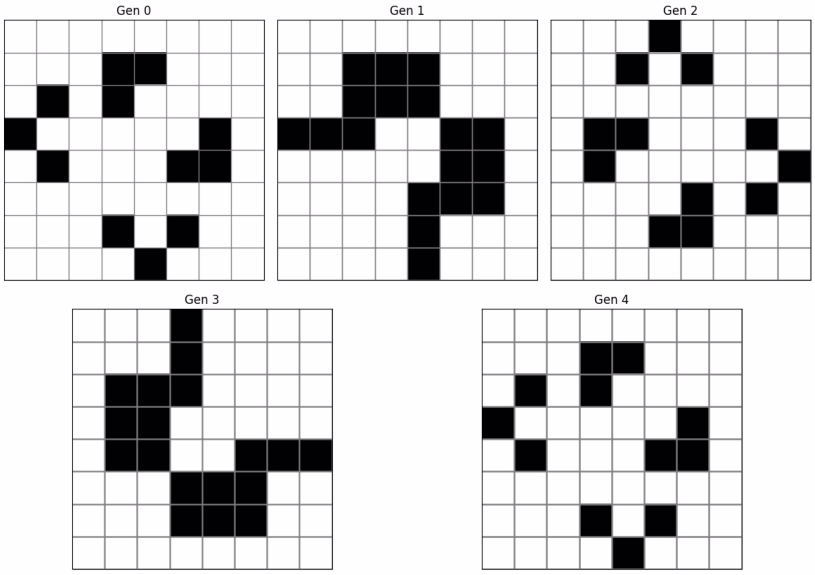

Figura 15: Patrón mazing: un oscilador de periodo 4 que oscila entre una forma cuadrada y una forma de cruz.

4.3. Naves espaciales

Las naves espaciales son patrones que se trasladan a través de la cuadrícula mientras oscilan. El ejemplo más conocido es el glider que se desplaza diagonalmente a través de la cuadrícula mientras oscila entre cuatro estados diferentes como se puede ver en la figura 16.

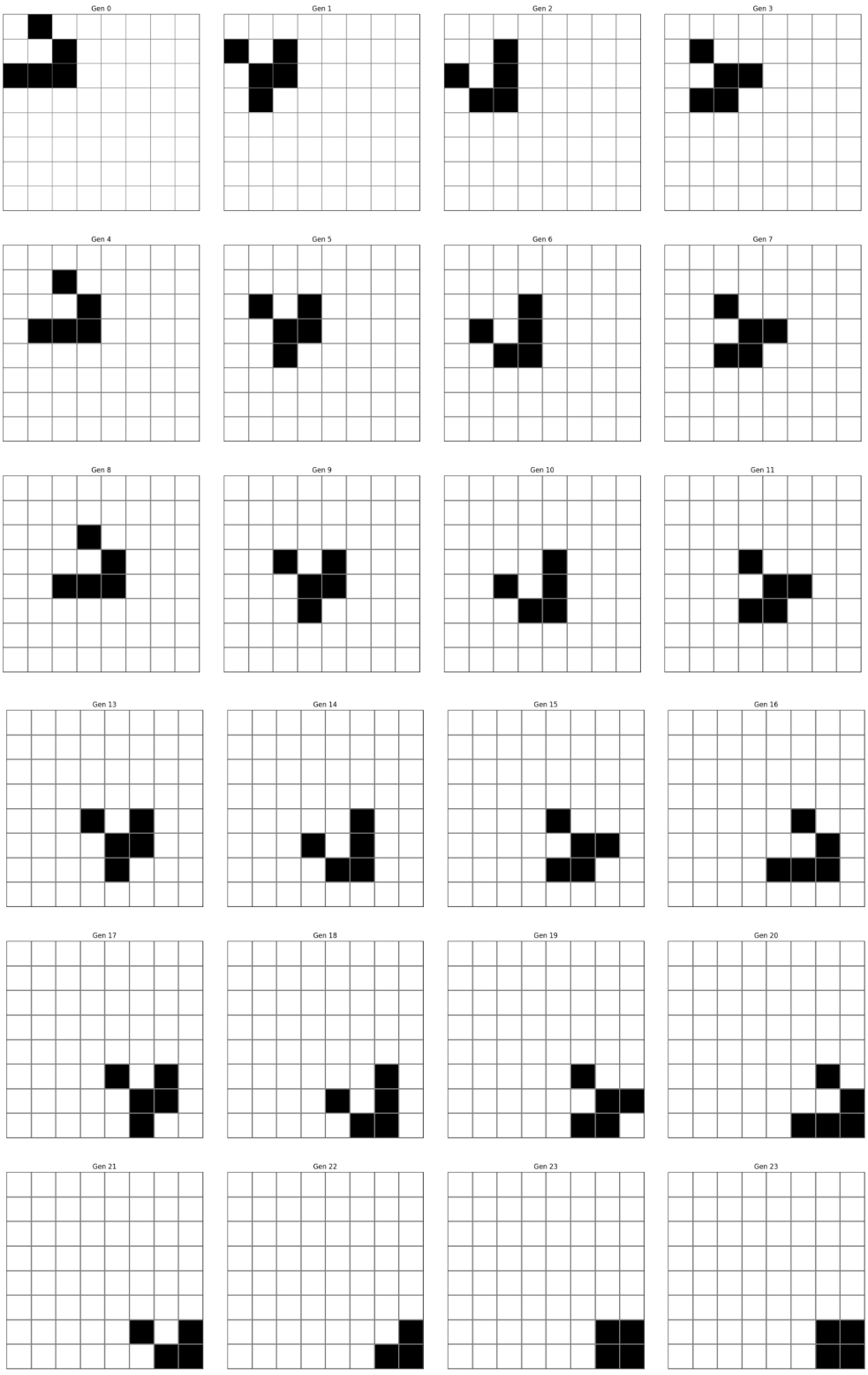

Figura 16: Patrón glider, generaciones 0-23: desplazamiento diagonal

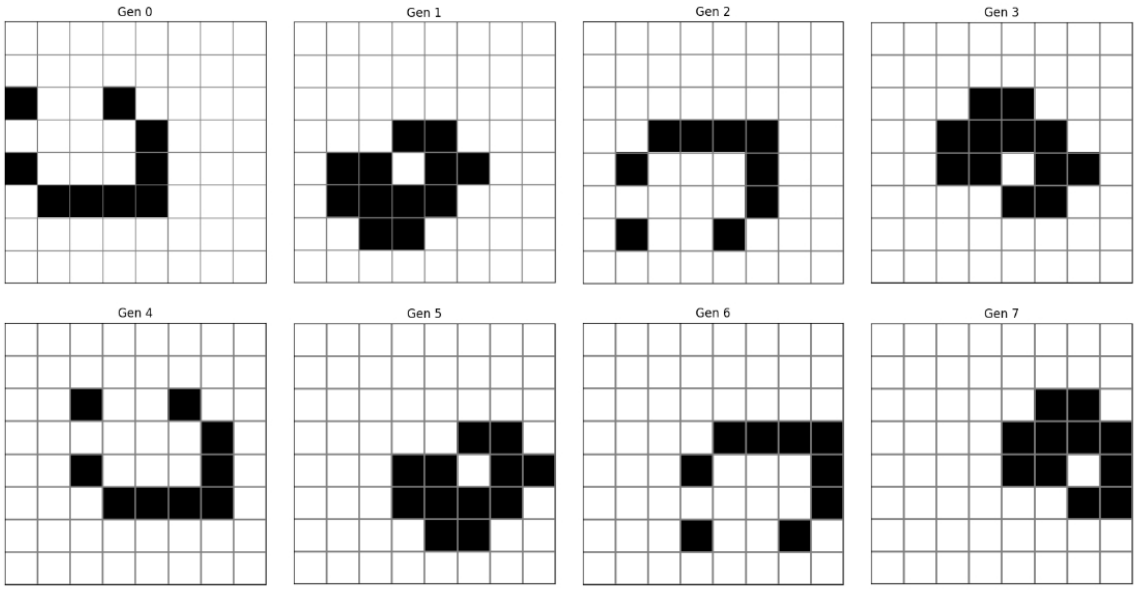

Figura 17: Patrón LWSS (Nave Ligera de Espacio Medio): una nave espacial que se desplaza horizontalmente mientras oscila en una secuencia de cuatro estados.

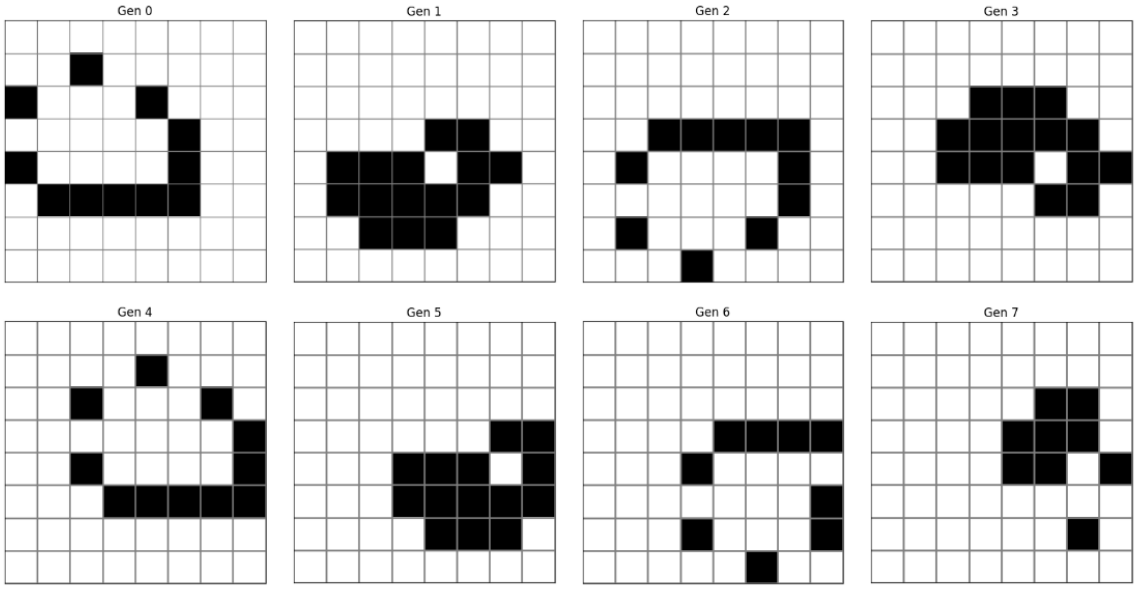

Figura 18: Patrón MWSS (Nave de Peso Medio): se desplaza horizontalmente por el tablero, el tamaño intermedio entre las naves espaciales LWSS y HWSS.

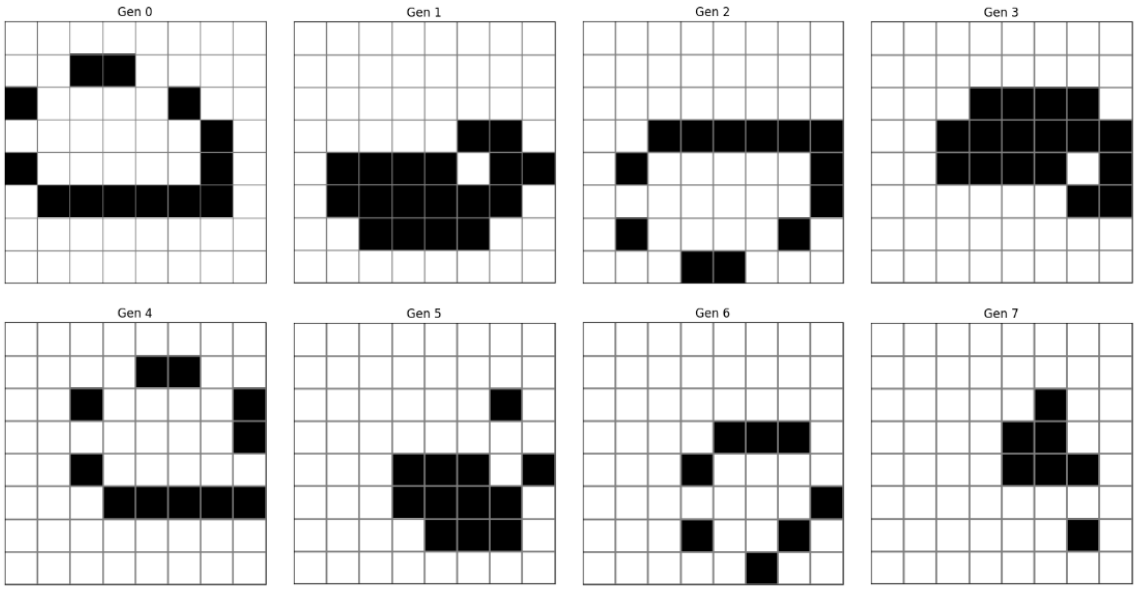

Figura 19: Patrón HWSS (Nave Pesada del Espacio): es la más grande de las tres naves espaciales básicas, se desplaza horizontalmente por el tablero hacia la derecha infinitamente.

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 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 ”construir” patrones adicionales, y otros pueden incluso comportarse como máquinas repliadoras.

Algunos patrones complejos son capaces de "disparar"naves espaciales, actuando como una especie de cañón. Un ejemplo famoso de este tipo de patrones es el "Gosper Glider Gun". Fue descubierto por Bill Gosper, el mismo genera un glider cada 30 generaciones que se desplaza indefinidamente a través del tablero como se puede ver en la figura 20.

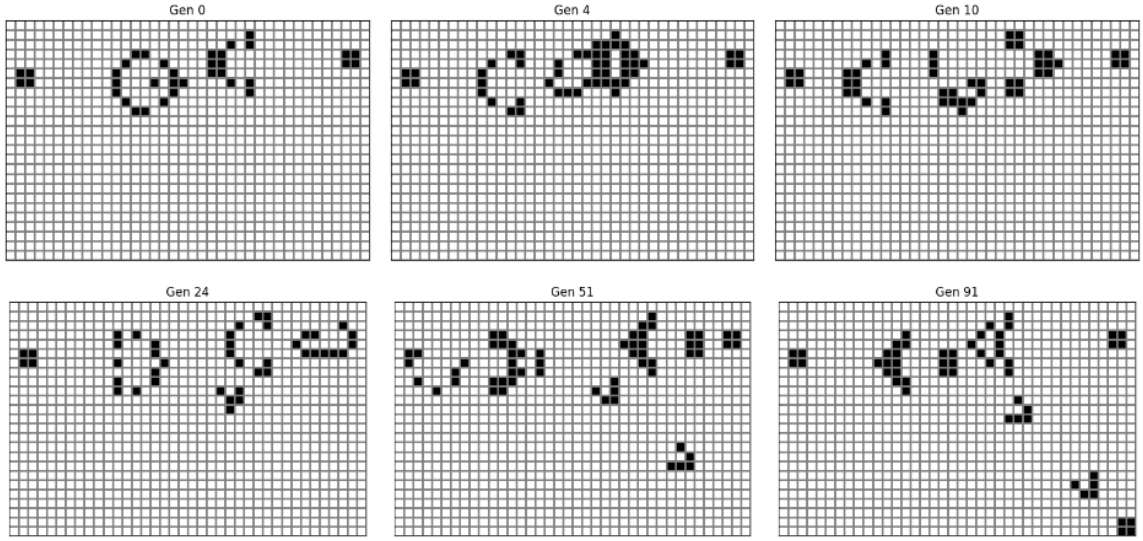

Figura 20: Patrón Gosper Glider Gun: genera un glider cada 30 generaciones.

Por otro lado, existe el "Pulsar", que si bien es un oscilador de periodo 3, para poder representarlo se necesita mínimo un tablero de 15x15. Está compuesto por un arreglo de células que "pulsan" entre tres configuraciones distintas. Esto se puede ver en la figura 21.

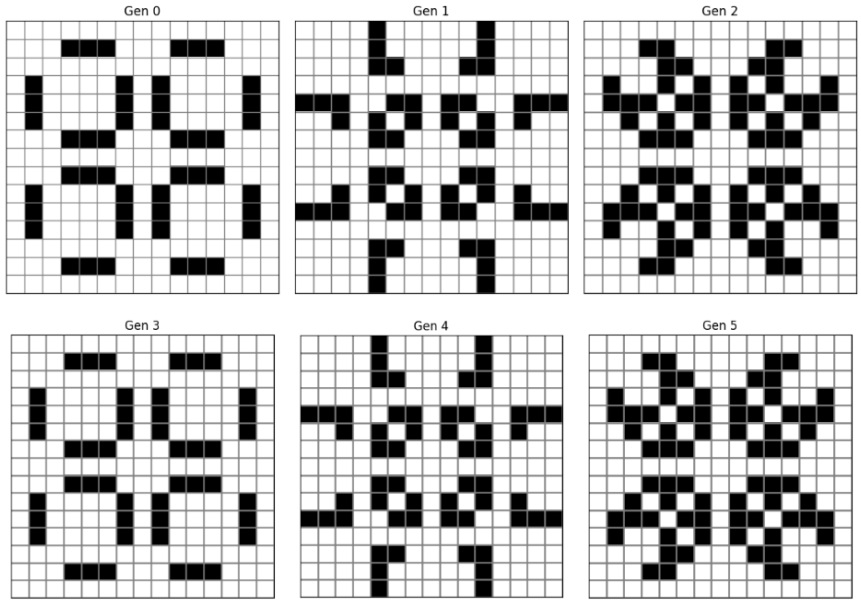

Figura 21: Patrón Pulsar: oscilador de periodo 3.

4.6. Cambiando las reglas

Como se menciono anteriormente, el Juego de la Vida se puede jugar con una variedad de reglas diferentes. Un patrón particularmente interesante es el "*Replicator*". Descubierto por el matemático

británico David Bell, es un patrón estático que se replica a sí mismo cada 15 generaciones. El Replicator solo aparece en el Juego de la Vida con las reglas de *HighLife (23/36)*, que son similares a las reglas originales de Conway, pero con la adición de que las células muertas con 6 vecinos vivos también nacen.

En la Figura 22 se puede observar algunas generaciones , asi como también en la figura 23

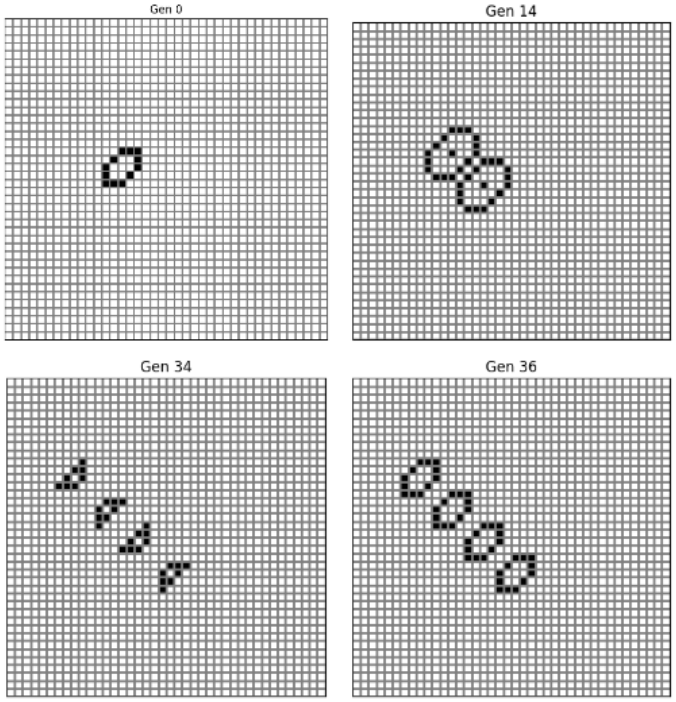

Figura 22: Patrón Replicator en un tablero 40x40, desde gen 0 a 36.

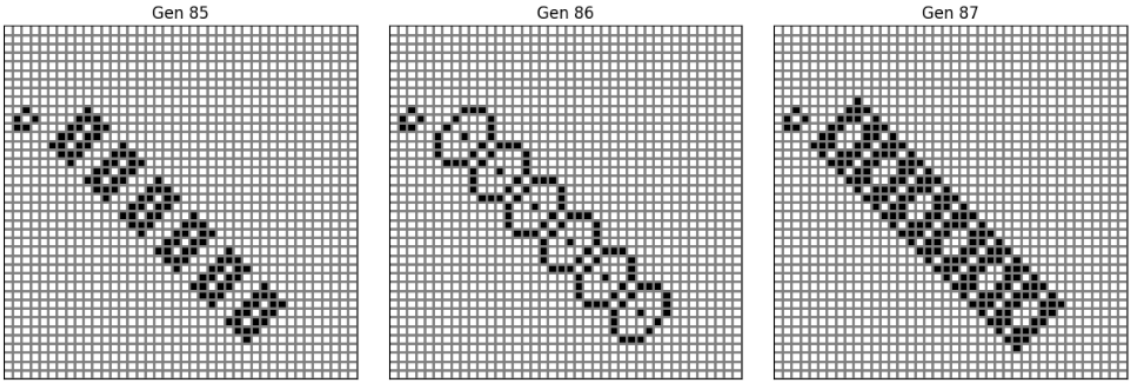

Figura 23: Patrón Replicator en un tablero 40x40, gen 85 a gen 87.

Una de las ventajas del Juego es su flexibilidad. Si bien las reglas originales generan una asombrosa variedad de patrones, cambiar las reglas puede resultar en nuevas dinámicas.

5. Anexo: Guía de Uso para Settings GoL

Settings GoL es una herramienta de interfaz gráfica de usuario (GUI) diseñada para facilitar la configuración y personalización del Juego de la Vida en PowerDEVS. Esta herramienta, implementada en Python, proporciona una serie de funcionalidades útiles para los usuarios. A continuación, se describen en detalle las funcionalidades de las pestañas de la herramienta.

5.1. Pestaña 1: Configuración del Proyecto

Esta pestaña proporciona dos botones principales: "Setear PowerDEVS" y "Restablecer a la versión original".

- **Setear PowerDEVS**: Este botón permite configurar PowerDEVS con los archivos necesarios para ejecutar el Juego de la Vida. Al hacer clic en este botón, la herramienta copia automáticamente los archivos necesarios en las ubicaciones correctas de la instalación de PowerDEVS en su sistema.
- **Restablecer a la versión original**: Este botón permite revertir cualquier cambio realizado por la herramienta en la instalación de PowerDEVS. Al hacer clic en este botón, la herramienta elimina cualquier archivo que haya copiado en las ubicaciones de PowerDEVS y restaura cualquier archivo que haya modificado a su estado original.

Figura 24: Pestaña 1: Configuración del Proyecto.

5.2. Pestaña 2: Creación de un Estado Inicial

Puesto que la creación de patrones en este formato puede ser un proceso laborioso, especialmente para tableros de grandes dimensiones, para crear patrones de manera visual y exportarlos en el formato correcto, esta funcionalidad se implementó en la pestaña 2 de la herramienta.

Esta pestaña permite crear un estado inicial personalizado para el Juego de la Vida. La pestaña consta de varios componentes:

- **Creación de la Cuadrícula**: En esta sección, el usuario puede definir las dimensiones de la cuadrícula del juego. El usuario puede ingresar el número de filas y columnas que desea para su cuadrícula en los campos de texto proporcionados. Al hacer clic en el botón "Crear Cuadrícula", la herramienta genera una cuadrícula con NxM celulas muertas de las dimensiones especificadas.
- **Configuración de las Células**: Una vez que la cuadrícula ha sido creada, el usuario puede configurar el estado inicial de cada celda haciendo clic en ella. Un clic cambia el estado de una celda entre vivo (representado por una casilla negra) y muerto (representado por una casilla blanca).
- **Configuración de las Reglas**: En esta sección, el usuario puede definir las reglas de supervivencia y nacimiento para el Juego de la Vida. El usuario debe ingresar las reglas en los campos de texto proporcionados.

- **Exportar Estado**: Una vez que el usuario ha configurado el estado inicial y las reglas del juego a su satisfacción, puede exportar esta configuración a un archivo de texto haciendo clic en el botón "Exportar Estado". La herramienta generará automáticamente un archivo de texto con la configuración actual en el formato correcto.

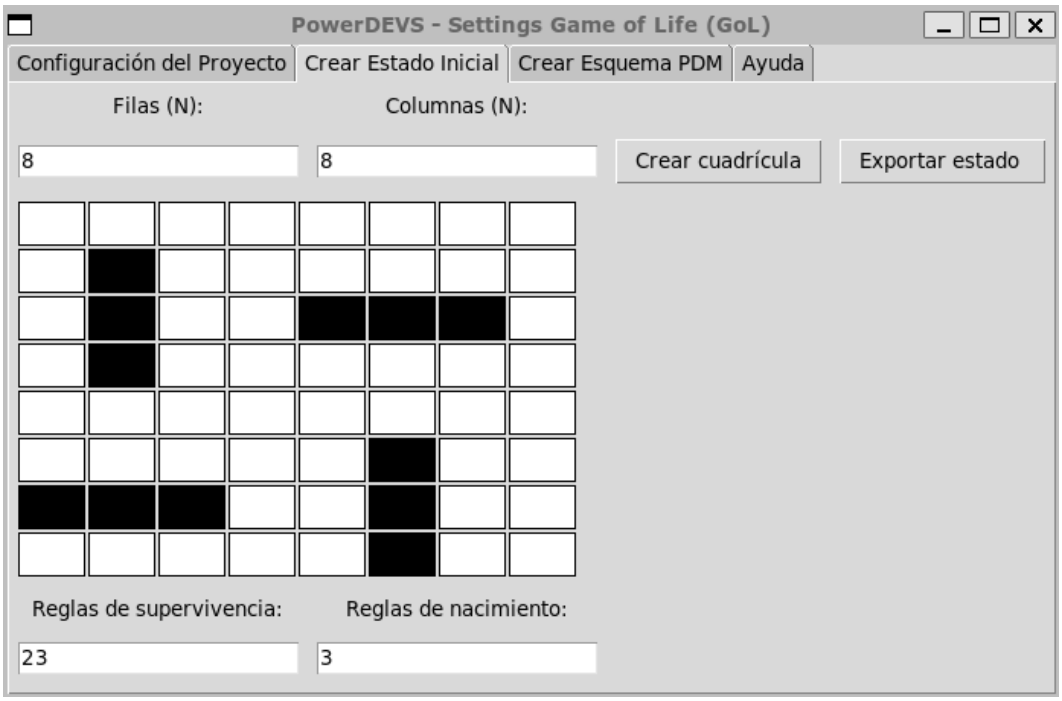

Figura 25: Pestaña 2: Creación de un Estado Inicial.

5.3. Pestaña 3: Creación de Esquemas PDM

Crear las conexiones entre células y mediator es todo un desafio a medida que el tablero crece en dimensiones. Para facilitar este proceso, la herramienta proporciona una funcionalidad para generar automáticamente esquemas PDM de dimensiones NxM listos para ser ejecutados en PowerDEVS.

El usuario puede ingresar las dimensiones deseadas en los campos de texto proporcionados y hacer clic en el botón "Generar PDM". La herramienta generará automáticamente un archivo PDM con un esquema de las dimensiones especificadas, listo para ser cargado en PowerDEVS.

El esquema generado incluirá todas las células necesarias, el mediator y las conexiones entre ellos. Esto facilita la creación de tableros de juego personalizados, ya que el usuario no necesita crear y conectar manualmente cada célula en PowerDEVS. Lo único es que va a tener que setear el estado inicial y archivo de salida en el mediator.

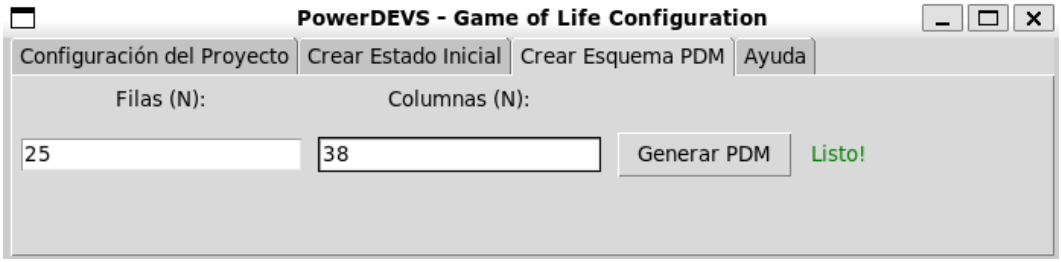

Figura 26: Pestaña 3: Creación de Esquemas PDM.

Nota: El archivo PDM generado se guarda en la carpeta examples/gol de PowerDEVS.

Con estas funcionalidades, Settings GoL proporciona una forma sencilla y eficiente de configurar y personalizar el Juego de la Vida para su ejecución en PowerDEVS.