Juego de la Vida

Trabajo práctico para Simulación

D'Autilio Joel, Rossi Pablo

1. Especificación DEVS

1.1. Idea intuitiva

El modelo DEVS que representa al juego de la vida es un acoplado de células que se comunican entre sí. Cada célula tiene un estado que puede ser vivo o muerto, y se actualiza en cada paso de la simulación. Las células se comunican con sus vecinas para conocer sus estados y así poder calcular el suyo.

Llamese I al tiempo entre estados del juego. Queremos que cada I segundos cada célula comunique su nuevo estado y se guarde el tablero. Para esto, siguen un ciclo de dos pasos que llamaremos *Comunicar* y *Actualizar* (fig. 1).

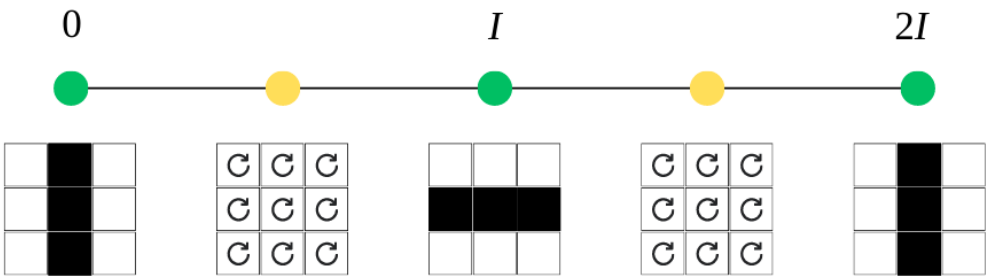


Figura 1: Ciclo de comunicación y actualización de una célula. En los puntos verdes, la célula comunica su estado a sus vecinas y se guarda el tablero. En los puntos intermedios amarillos, la célula actualiza su estado de acuerdo a las reglas del juego y al estado de sus vecinas.

- *Comunicar*: la célula comunica su estado actual a sus vecinas y guarda en una variable interna el estado de su vecindario. Ocurre cada I segundos comenzando por $t = 0$.
- *Actualizar*: la célula actualiza su estado de acuerdo a dicha variable y a las reglas del juego. Ocurre cada I segundos comenzando por $t = I/2$.

El fin de dividir el intervalo en dos subintervalos es separar las acciones de comunicar y actualizar para asegurar que cuando una célula calcule su nuevo estado todas sus vecinas le hayan comunicado el suyo.

1.2. Formalización

El DEVS que representa a una célula está definido como

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

donde

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

  La entrada es un par (id, s) que indica que el vecino con ese id nació ($s = 1$) o murió ($s = 0$).

- $Y = \big[(\mathbb{N} \times \{0, 1\}) \times \{0\}\big] \cup \{(0, 1)\}$

  La salida es de la forma $((id, s), 0)$ ó $(0, 1)$.

  Por el puerto 0 se emite el par (id, s) con el id y estado actual de la célula.

  El puerto 1 se usa para descartar la salida cuando el estado de la célula no cambió.

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

  El estado es una tupla $(id, s, vv, vc, pa, \sigma)$ donde

  - $id \in \mathbb{N}$ es el identificador de la célula.
  - $s \in \{0, 1\}$ es el estado actual de la célula.
  - $vv \in \mathbb{N}$ es la cantidad de vecinos vivos.
  - $vc \in \mathit{Bool}$ indica si el vecindario cambió en el último paso.
  - $pa \in \{\mathit{Comunicar}, \mathit{Actualizar}\}$ indica la próxima acción a realizar.
  - $\sigma \in \mathbb{R}_0^+$ es el tiempo restante para realizar la próxima acción.

- $\delta_{\text{int}}((id, s, vv, vc, pa, \sigma)) = \begin{cases} (id, s, vv, vc, Act, I/2) & pa = Com \\ (id, s, vv, F, Act, I) & pa = Act \wedge \neg vc \\ (id, s, vv, F, Act, I) & pa = Act \wedge vc \wedge s = s' \\ (id, s', vv, F, Com, I/2) & pa = Act \wedge vc \wedge s \neq s' \end{cases}$

  donde $s' = \text{calcularEstado}(s, vv)$.

  En este punto, pa es la acción a realizar en el instante actual. Si ésta es *Comunicar*, se reinicia el reloj interno y la próxima acción será *Actualizar*.

  Si la acción es *Actualizar* y el próximo estado es el mismo que el actual (casos 2 y 3) se evita la acción de *Comunicar* estableciendo el reloj en I y la $pa = Actualizar$.

  Por último, si el próximo estado es distinto del actual (caso 4), se establece la próxima acción en *Comunicar* y se reinicia el reloj interno.

- $\delta_{\text{ext}}((id, s, vv, vc, pa, \sigma), e, (id, x)) = (id, s, vv', T, pa, \sigma - e)$

  donde $vv' = \begin{cases} vv + 1 & x = 1 \\ vv - 1 & x = 0 \end{cases}$

- $\lambda((id, s, vv, vc, pa, \sigma)) = \begin{cases} ((id, s), 0) & pa = Comunicar \\ (0, 1) & pa = Actualizar \end{cases}$

  La célula produce una salida sólo en el caso de que la acción sea *Comunicar*; si la acción es *Actualizar* no ocurre una salida. Para simular esto, utilizamos el puerto 1 como descarte.

- $ta((id, s, vv, vc, pa, \sigma)) = \sigma$

Consideramos a los parámetros $I \in \mathbb{R}^+$, que representa el tiempo entre estados del tablero, y $RS, RN \subseteq \mathbb{N}$ que representan a las reglas de supervivencia y nacimiento respectivamente. También utilizamos la función calcularEstado() que es el próximo estado de una célula dado su estado actual y el número de vecinos vivos.

$$\text{calcularEstado}(s, vv) = \begin{cases} 1 & s = 1 \land vv \in RS \\ 1 & s = 0 \land vv \in RN \\ 0 & \text{en otro caso} \end{cases}$$

2. Implementación en PowerDEVS

Una célula del tablero se implementa a través de un modelo DEVS atómico que incluye un parámetro para el id, una entrada y una salida, como se puede ver en la figura 2.

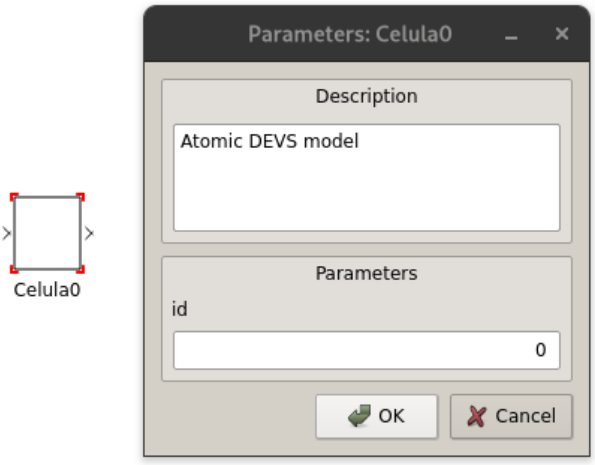
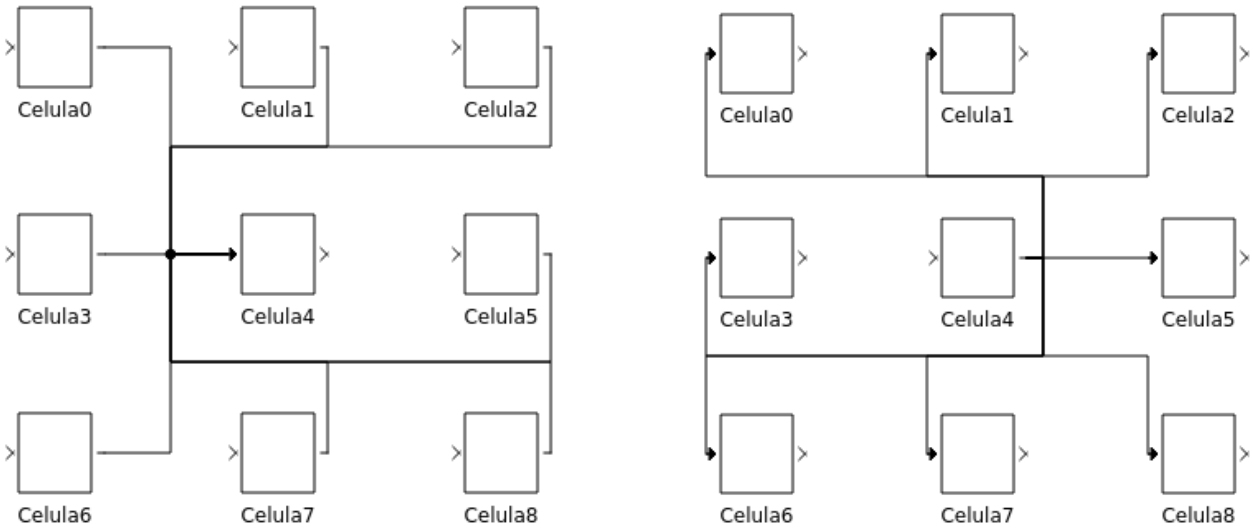


Figura 2: Modelo atómico de una célula.

Una célula se comunica con sus vecinas a través de sus puertos de entrada y salida, como se observa en la figura 3. Por su puerto de entrada, a lo sumo 8 vecinas le comunican sus cambios de estado (fig. 3a) y por su puerto de salida les comunica cuando cambia el suyo (fig. 3b).

En la figura 4 se observa un tablero completo de 4×4 con las conexiones mencionadas anteriormente. Además, cada célula se comunica con un DEVs atómico 'Escritor' que se encarga de escribir los resultados de una simulación en un archivo.



(a) Conexiones de entrada.

(b) Conexiones de salida.

Figura 3: Conexiones de una célula con sus vecinas.

2.1. Flujo de evolución

Cada célula, obtiene su estado inicial, las reglas de supervivencia y nacimiento, y el tiempo entre estados, a través de un archivo de configuración. De la misma manera, el Escritor obtiene el estado inicial del tablero y el tiempo entre estados del mismo archivo.

La simulación consta de dos etapas: una de *Comunicación* y una de *Actualización*. En la primera, cada célula comunica su estado a las vecinas y se escribe en un archivo de salida el estado actual del tablero. En la segunda, cada una actualiza su estado.

Para optimizar este proceso, se lleva una variable $vc \in Bool$ que indica si el vecindario cambió desde la última actualización. Si una célula está en la etapa de Actualización y $vc = F$ entonces no es necesario actualizar su estado, y por lo tanto tampoco es necesario comunicarlo a sus vecinas. Además, si el vecindario sí cambió ($vc = T$) pero el estado resultante es el mismo, tampoco es necesario comunicarlo. En estos casos, la célula no pasa por la etapa de Comunicación hasta que su estado cambie.

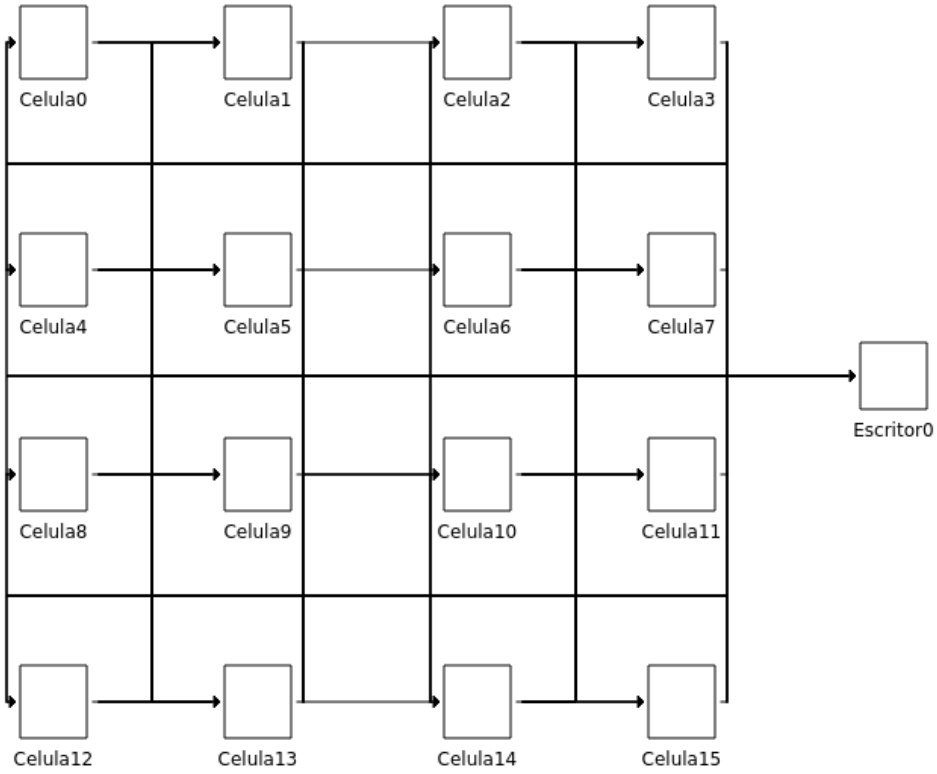


Figura 4: Tablero de 4×4.