# *Conway's Game of Life*

**Edgar Andrés Margffoy Tuay** 201412566

Universidad de los Andes Concurrency, Parallelism and Distribution

28 de noviembre de 2018

## 1. Implementation Details

Due to its local behaviour, the Conway's Game of Life can be approached as a parallel problem, on which, each update to each individual cell does not depend on the update result of the other cells. Moreover, a Game of Life board can be seen as a collection of sub-boards that also are instances of the Game of Life, and therefore it represents a parallel recursive function that can be implemented via Fork-Foin.

Formally, given a board of size $m \times n$, we take four sub-boards of size $m/2 \times n/2$ and apply recursively the Life game board partition, until the resulting board dimensions are $m_c \times n_c$, with $m_c$ and $n_c$, being the cutoff height and width, respectively. Then the B3/S23 rule is applied over the cutoff board. When all four processes finish, the parent process is on charge of merging all the subarrays into a single board. An important note about this process, is that cells that lie on a board boundary, depend on the state of cells that are across the boundary on other boards. To overcome this issue, for each sub-board, the partition implementation takes into account ghost cells that keep track of the cells that are on other boards, but are not to be updated after applying the rules to the sub-board. Figure 1 depicts the process. To run an example board, please execute `LifeCLI.example` on the `iex` interpreter.

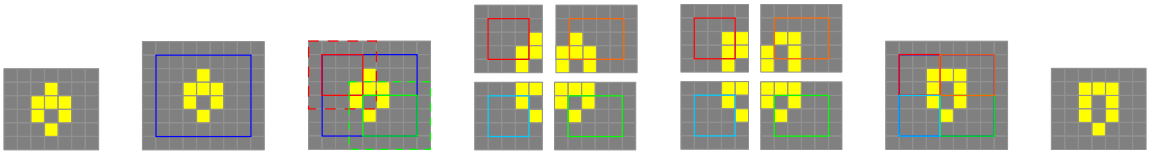


*Figura 1:* Parallel Conway's Game of Life: First, given a board, padding is added such that boundary cells can be updated. Then, the active area is split into four subarrays and assigned to a parallel process, each one of them contain additional columns and rows that allow to update their corresponding boundary cells. When the array dimensions are lesser or equal to the cutoff value, the active area cells are updated in parallel. Finally, the parent process assembles the final board using the resulting active areas of each worker.

### 1.1. Parallel Complexity

Due to the independence between concurrent cell updates, and given enough resources, the process can be parallelized up to $mn$ processors. Given that the work $T_1$ elapsed by a single processor corresponds to $\mathcal{O}(mn)$, the complexity of the parallel work $T_P$ corresponds to $\mathcal{O}(1)$. With respect to the span of the parallel process $T_\infty$, this measure is bounded by the costs of splitting and merging arrays, which in the worst case correspond to $\mathcal{O}(m+n)$ (assuming constant array access cost and constant array merging time). Thus, the total complexity for the parallel version of Conway's Game of Life corresponds to $\mathcal{O}(m+n)$.

- Span: $\mathcal{O}(m+n)$
- Work: $\mathcal{O}(1)$