Nathan Larson
CSC 412

Assignment 04 Report

Implementation

For version 1, I decided to implement a secondary semaphore lock on top of the grid mutex lock controlling threads grid access. It starts in a for loop in the main function that creates all of the determined number of threads and assigns them a number of rows to handle. Inside of each thread, there is a while loop that continues looping in order for the threads to continue looping until the program is ended. Each first first attempts to acquire the mutex lock, then once acquired compute and assign the next generation for the rows it has been assigned. The thread then attempts to get the semaphore lock using sem_wait. After then incrementing the swapCounter value that controls how many threads have ended their critical section, the thread checks if it was the last thread to execute (swapCounter == numThreads) which would then lead to the grids being swapped and swapCounter reset. The thread would then release both locks and continue operation. With this solution all the threads dont necessarily execute in order, but all threads execute only once per swapgrids call.

For version 2, I creates a 2D array of mutex locks, declaring and instantiating in almost the same spots as the regular 2D grid array for readability. Each thread would then get a random location on the grid, attempt to acquire all 9 of the block in that radius, then calculate what the next generation of that block should be and swap it out in real time.

Difficulties

A difficulty in version 1 was the way that I was going to have the threads check and handle whether or not the next grid was ready to be swapped out after being fully calculated. I thought about doing an array of 1 and 0 to control whether or not all threads have finished or not, but it ended up being much simpler to just implement a semaphore lock and count to control these threads.

In order to keep the regular implementation and the extra credit part separate, I have a separate folder named ExtraCredit that contains a copy of version 1 and the extra credit version of the interpreter, which acts as a command line interpreter and can launch multiple instances of a cell automation at once.