439 — Thy Comp — TM Cella Rule 150

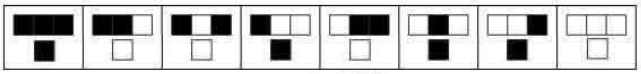
Project #2 – TM Cella Rule 150

Introduction

This project is to write a program to simulate a Turing Machine (TM) which in turn simulates and displays the generational progress of Wolfram's Rule-150 cellular automaton. (You are already familiar with the Wolfram cellular automata.) The program will be written in Javascript with an HTML web page for display. The Tm will be runnable at full speed, and also under user control stepping one new-generation cell being written at a time.

Rule-150

Wolfram's Rule-150 is based on a 1D array where each cell is "active". See his book or Wikipedia for more info. Rule-45 looks like this:



NB, generations #0 thru #3 should look like this (roughly, using black for state = 1 and white for state = 0, and this is only the center area of the first 4 rows).

Display Setup

For displaying the cellular automaton, your program should initialize a 41x20 square grid to have all cells empty. Then turn the top row's 20th (center) cell on. This represents the initial (#0 == seeded) generation. This grid represents your TM's 2D tape. Include your team name on the web page above the grid.

For displaying the TM state changes, your program should include a 2D state-array whose slots are each filled with one of your TM's states. The TM start state's slot should have its state highlighted. (Yes, this is yet another grid, but we will call it an array to avoid confusion.)

The TM tape Head should initially be "over" the seeded cell. To make this clear to the user, the cell the Head is over should always be highlighted.

Turing Machine

Your TM will use the cell grid as its own 2D tape, with one read/write tape Head. (The 2D effect merely simplifies/speeds the Head movement, but provides no extra computational power.) The Head can move UDLR (Up, Down, Left, or Right). The Head can read or write the contents of a tape cell. Your TM will need a "program" to create the next cellular generation from the previous one, by looking at one previous cell (in the grid row above) at a time.

Sample Action

Write-Next-Cell

Write next new-gen row's cell, directly below the tape Head.

Pre-condition: Head at old-gen row's "middle-context" cell, ready to write the cell below it.

TM Actions:

o- Left to 1st cell of old-gen tri-cell context
o- Read old left cell, move Right
o- Read old middle cell, move Right
o- Read old right cell, move Left, back over old middle cell, again
--> Now you should know the value of the next gen "middle" cell
o- Down to new-gen row's "middle" cell
o- Write new gen middle cell, move Up to old row middle cell

o- Move Right, over to the next old-gen row's "middle" cell.
Post-condition: Head at next old-gen row's "middle-context" cell, ready to write the cell below it.
Note: you'll need to change the TM state as you recognize parts of the Rule-150 old-gen state context.
Also, this is a sample. You do not have to implement the TM using this.

Complexity Order

You should prepare a 1-page (at most) paper describing your analysis of the Big-O running time of your TM algorithm used to build 1 new generation of the Rule-150 automaton. Address the usual issues such as main operations, input size, etc.

Team

The team size is the same as before, but you can change team members from the previous project if you wish.

Project Reports

As before, but simplified a bit. Please check out the updated sample Report pdf.

Readme File

As before.

Academic Rules

Correctly and properly attribute all third party material and references, if any, lest points be taken off.

Submission

As before.

Grading

As before.