SYSC 5104 - METHODOLOGIES FOR DISCRETE EVENT MODELLING AND SIMULATION

Prof. G. Wainer

Assignment 2:

Real-Life System Selection and Implementation Using Cell-DEVS Modeling

Joe Huntley

MEng (Electrical and Computer)

101059477

13 Nov 2020

Selection of a Real Life Model to Emulate Using CellDEVS

A Little on Cellular Automata:

Cellular Automata (CA) are discrete, abstract computational systems that may be used in a variety of scientific fields [1]. According to Berto et al. [1], CA have the following four capabilities:

1. As powerful computational engines.
2. As discrete dynamical system simulators.
3. As conceptual vehicles for studying pattern formation and complexity.
4. As original models of fundamental physics.

In this report, one specific real system is identified that may be modeled by a CA. In [2], the authors claim that CA may be used, by applying the benefits of capabilities 1-3 above, to model pedestrian traffic in an open space. Pedestrian traffic flow is a good example of using CA to model a real life system as several parameters may be adjusted in order to most accurately emulate real life, such as speed, desired outcome (shortest path vs most "scenic" route) as well as the desire to avoid conflicts (bumping into other people or avoiding them as much as possible, like during a pandemic). Additionally, this model may be tailored to resemble an open area such as an airport terminal, or a somewhat closed system with predetermined walkways such as a downtown core.

Description of Model Behaviour from the Article:

To most closely follow the system described in [2], the model that will be implemented in part 2 of this assignment will emulate an open area occupied by varying numbers of "pedestrians" emerging from up to 4 different directions. In [2], this system is described as the "Grand Central Station Problem". The goal of modeling this system will be to "determine the essential factors that inform pedestrian decision making and to create a rule set that results in realistic emergent group behavior" [2].

The "Local Rule Set" [2] is described as follows: a floor area of a matrix of cells '(i, j)' will have a number 'N' of pedestrians enter at each time step 'T', and proceed to the opposite side. Origin '(i_o, j_o)' and destination '(i_d, j_d)' cells will be determined by a random distribution. The pedestrian will attempt to reach their destination by following local rules, such as avoiding collisions or conflicts. At each timestep 'T', each pedestrian attempts one step forward, and if that step is blocked, they will attempt a sidestep. If the intended cell is occupied during a sidestep, the pedestrian will "bump" the occupant into a sidestep relative to its forward movement. If that pedestrians intended sidestep cell is blocked, they will bump that occupant. This continues until a free cell is found or a pedestrian gets bumped off the floor matrix. A visual depiction of pedestrian movement in four timesteps is shown in [2, Fig. 1], and is provided in Fig. 1 below:

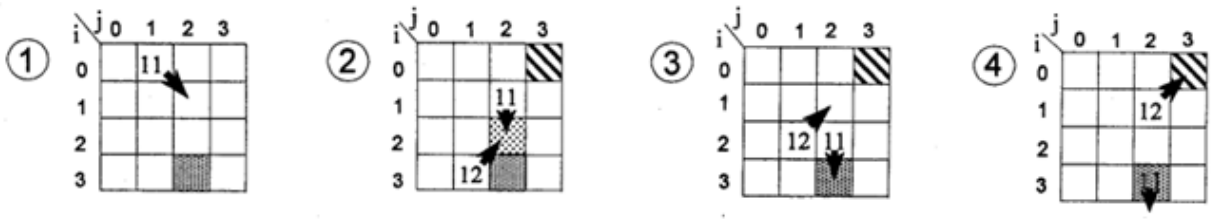


Fig. 1. Illustration of pedestrian movements in four consecutive time steps

Continuing the discussion of the local ruleset, an entity enters the matrix at time T_i, at an origin cell (i_o, j_o), and is assigned and entity number 'e_n' until it reaches its destination cell (i_d, j_d). At timestep T_{i+1}, entity e_n moves towards its destination; if $j_d - j_o = 0$, the entity moves towards the cell directly ahead ($i_o + 1$, j_o). If that cell is occupied, e_n will "adjust" its path by choosing either diagonal cell ahead ($i_o + 1$, $j_o \pm 1$) without going of the matrix. If all three attempted cells are occupied, e_n will attempt to sidestep.

When $j_d - j_o = 0$, e_n will attempt to sidestep by choosing with equal probability either cell to the side ($i_o + j_o \pm 1$), without going off the matrix. If that cell is occupied, e_n will bump the occupant.

If $j_d \neq j_o$, e_n will attempt to move to a cell forward and diagonal of its current position: ($i_o + 1$, $j_o \pm 1$). If that cell is occupied it will attempt to occupy the cell directly ahead ($i_o + 1$, j_o). If that cell is occupied, it will attempt to sidestep by choosing the cell to its side which is closest to its destination (i_o, $j_o \pm 1$) without going off the matrix. If that cell is occupied, it will bump the occupant.

When bumping, the bumped entity moves to the side closest to the destination, or arbitrarily to either side if $j_d - j_o = 0$. Only a bumped entity may leave the matrix. This entity will itself bump another occupant if the desired cell is occupied. The bumping algorithm is recursive, and may result in several bumps before a vacant cell is found or an entity is bumped from the matrix.

Implementation of Selected Model in CellDEVS

Model Behaviour Variation from Article Description:

Due to the intended scope of this assignment, not all of the functionality described in the article has been implemented. For instance, the "pedestrians" do not enter the grid from all directions, but simply are initiated to begin at the top, and work there way down to the bottom of the non-wrapped grid space. Also, the model is not currently configured to avoid bumping into other pedestrians, but only to avoid static obstacles it may encounter in the grid.

Formal Specification:

The specification for an atomic, asynchronous Cell-DEVS is described as follows:

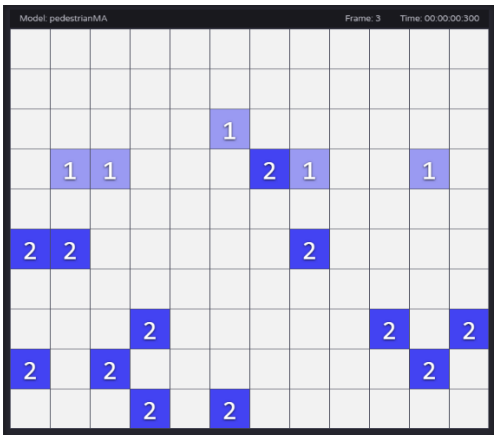
Timed Cell DEVS (TDC) = <X, Y, S, Θ, *I*, N, d, τ, δ_{int}, δ_{ext}, λ, D>

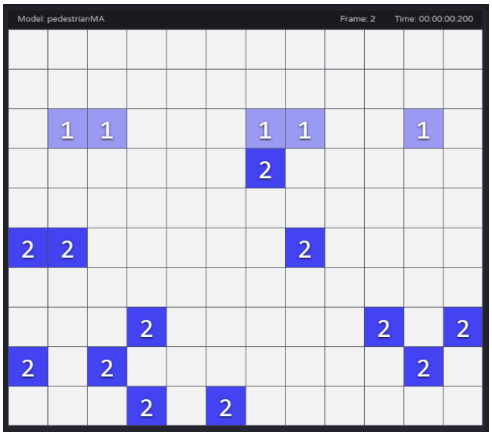
For the purposes of this assignment, students were directed to focus solely on parameters “d” (delay of the cell) and “τ” (Tau, the local computing function), as all other parameters and functions are provided within the provided simulation architecture. For the pedestrian model, the delay was 100ms. For the Tau function, the following rule-set shown in Fig. 2 contained within the code for model file “pedestrianMA.ma” was implemented:

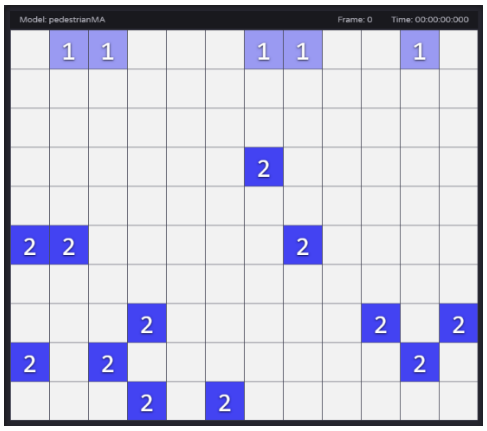
```
[pedestrian-rule]

%cells moving down (right)


%If cell is occupied and cell in front is not, then "move" (vacate cell)

rule : 0 100 { (0,0) = 1 and (1,0) = 0 }

%If cell is empty and cell behind is occupied, then "move" into (occupy cell)


rule : 1 100 { (-1,0) = 1 and (0,0) = 0 }


%cell has an obstacle, pedestrian must move right or left:


%If cell to bottom left is obstacle, and adjacent cell is occupied, move into 'this'
cell

rule : 1 100 { (0,0) = 0 and (0,-1) = 1 and (1,-1) = 2}

%If cell to bottom right is obstacle, and adjacent cell is occupied or an obstacle,
move into 'this' cell

rule : 1 100 { (0,0) = 0 and (0,1) = 1 and (0,2) >= 1 and (1,1) = 2}




%Default rules:

%If completely blocked, stay put

rule : 1 100 { (0,0) = 1 and (0,-1) = 2 and (1,0) = 2 and (0,1) = 2}


%If cell is an obstacle, stay an obstacle

rule : 2 100 {(0,0) = 2}

rule : 0 100 { t }
```

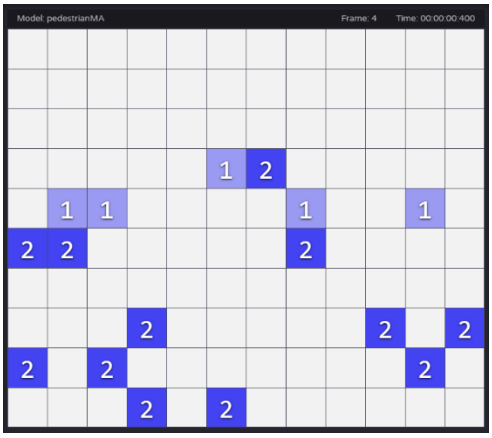
Fig. 2 Rule-Set Implemented for Pedestrian Model Behaviour

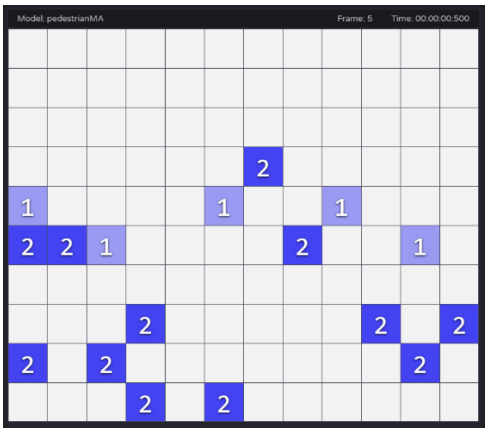
The set of rules shown in Fig. 2 resulted in the following behaviour, captured in a series of screen clippings, using the provided DEVS Web Viewer located at webpage https://www.omarhesham.com/arslab/webviewer/:

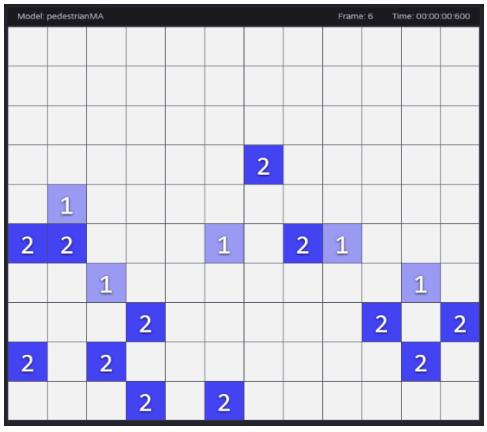

Model: pedestrianMA
Frame: 3
Time: 00:00:00:300

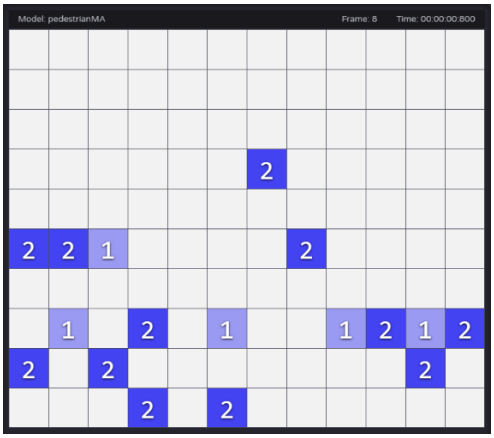

Model: pedestrianMA
Frame: 2
Time: 00:00:00:200

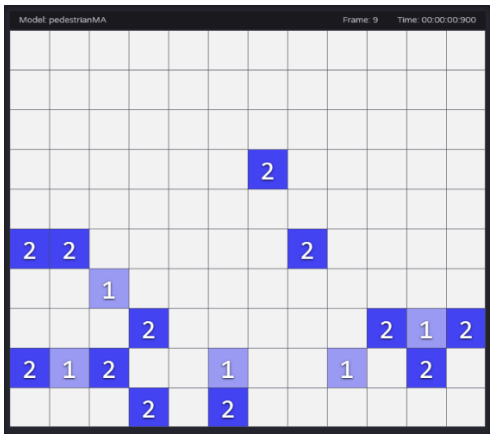

Model: pedestrianMA
Frame: 0
Time: 00:00:00:000

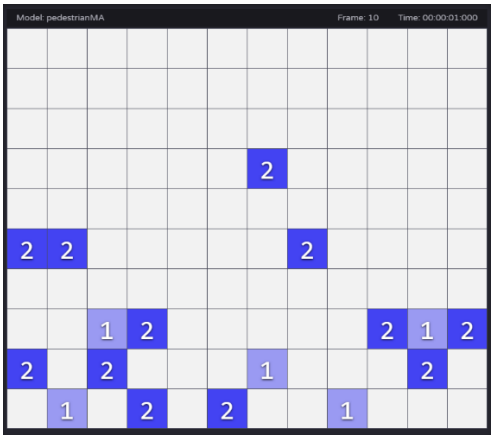

Model: pedestrianMA
Frame: 4
Time: 00:00:00:400

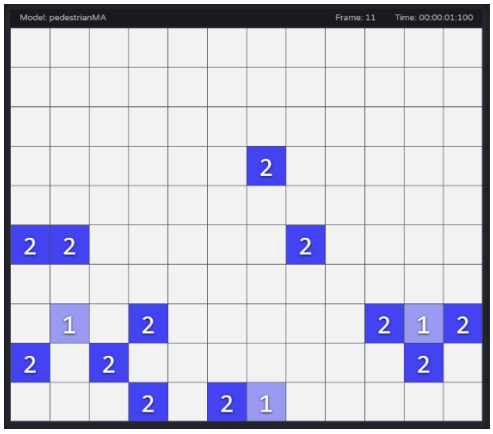

Model: pedestrianMA
Frame: 5
Time: 00:00:00:500

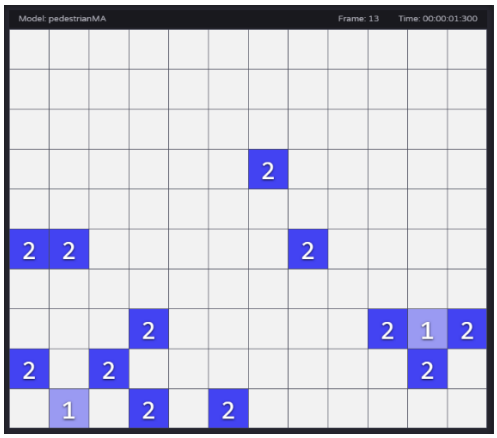

Model: pedestrianMA
Frame: 6
Time: 00:00:00:600


Model: pedestrianMA
Frame: 8
Time: 00:00:00:800


Model: pedestrianMA
Frame: 9
Time: 00:00:00:900


Model: pedestrianMA
Frame: 10
Time: 00:00:01:000





As shown by the series of clippings above, the "pedestrians" move from the top of the screen down to the bottom. If they encounter an obstacle, the first attempt to move to the right of the obstacle. If that is already occupied by another pedestrian or obstacle, the pedestrian will then attempt to move to the left. Once the pedestrian has no obstacle in their way, the continue their journey "southward". If a pedestrian encounters an obstacle in front of them, but cannot move left or right due to more obstacles, they will enter a steady state of remaining in place.

An interesting side note is that the original suggested DEVS Web Viewer results in an odd display of behaviour where it seems the grid space and model locations in that space are rotated 90° counter-clockwise. Also, when the pedestrian's anticipated movement would be to the right according to the code (to move around an obstacle), it was observed to actually move to the left. It became evident to this author that the Web Viewer at http://ec2-3-235-245-192.compute-1.amazonaws.com:8080/devs-viewer/app-simple/ was misinterpreting the (i, j) coordinate system used to indicate neighbouring cells and resultant behaviour.

Other Experiments

Other layouts were tried in order to observe the behaviour of the model under various conditions. Using the model "pedestrianMA**2**.ma", a near-checkerboard pattern is implemented for the obstacles. The pedestrians themselves are still arranged in the original configuration. The resulting initial state of the grid is shown in Fig. 3 below.

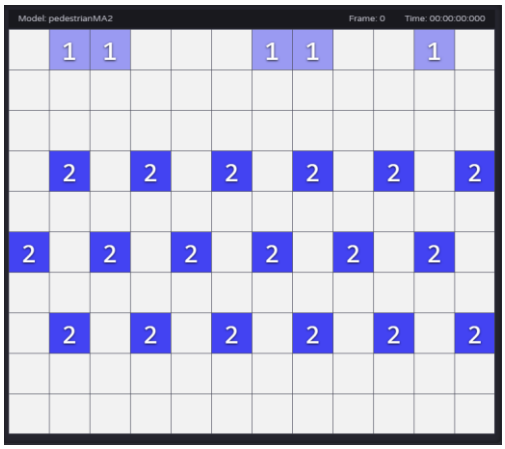


Fig 3 – Resulting initial configuration using "pedestrianMA2.ma" model file.

In the configuration shown in Fig 3, the models still behave as expected, but take only 1s300ms to clear the grid. In the original configuration, the models took 1s400ms to clear the grid (one extra cycle). The following shows a third configuration, where an entire row of pedestrians are initiated (this is the "pedestrianMA3.ma" file). The obstacles remain in the same configuration as they are in pedestrianMA2.ma.

In the configuration shown in Fig 4, only about half of the pedestrians make it to the bottom of the screen. Since no rules exist to cover the condition where the pedestrian is blocked by an obstacle in front, but there are neighbours occupying cells both to the left and right, half the pedestrians are dropped from the grid after the first line of obstacles (Fig. 5). Furthermore, after proceeding to the second row of obstacles, the pedestrian on the far right is dropped (Fig 6) since the border is not wrapped. The various experimental frameworks described in these paragraphs are a good indicator of the status of the behaviour of the model, and prove as an invaluable tool when attempting to have the code provide the intended model behaviour.

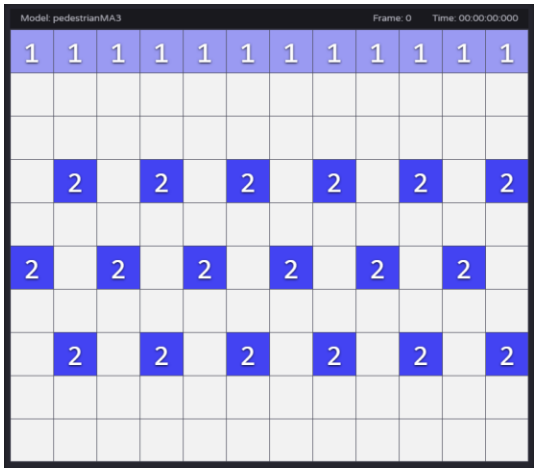


Fig 4 – Initial configuration with line of pedestrians

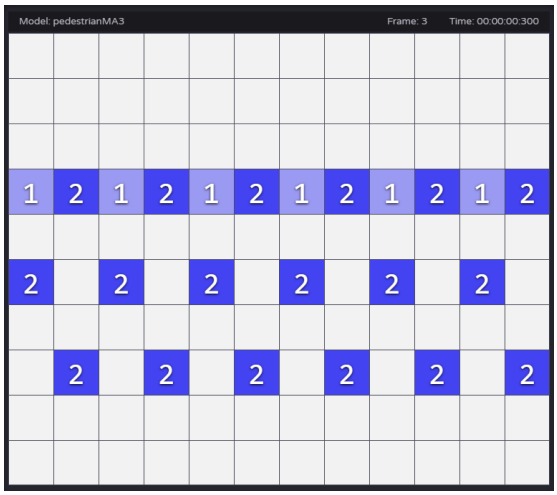


Fig 5 – Half of pedestrians lost due to lack of rule for this case

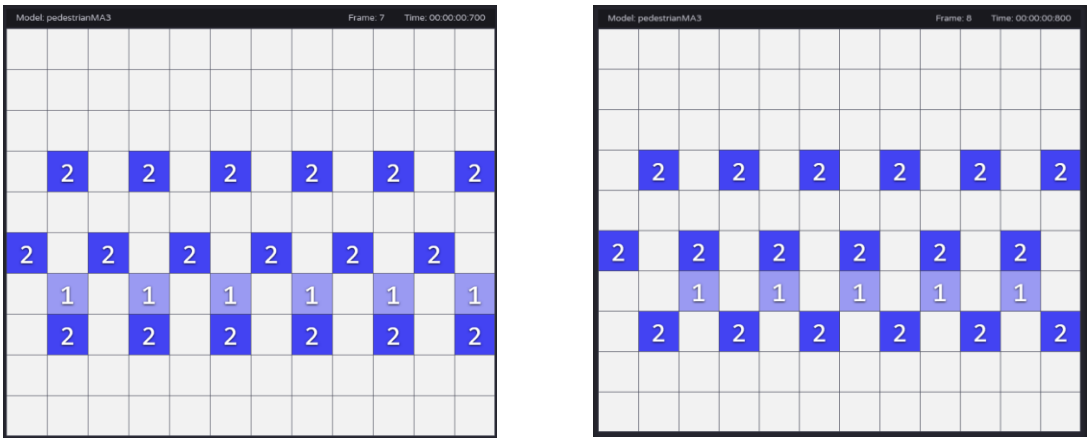


Fig 6 – Result of using a "no-wrap" border: pedestrian on far right is lost when shifting right.

Running the Simulation:

In order to run the simulation in the CellDEVS Viewer and be provided with a visual depiction of model behaviour, there are certain steps which must be taken. The first thing that must be done is to ensure that the provided "simu.exe" file and "cygwin1.dll" file are in the same folder as the *.bat file, as well as the model (*.ma) and log (*.log) files. Editing the *.bat file will allow the user to specify the model file to be used, as well as the log file to receive the output from running simu.exe. Once the simulation is run successfully (and a log file generated), these files need to be dragged and dropped to their corresponding places on the CellDEVS Viewer webpage. After clicking on "Parse Simulation Log", the simulation is ready to be run. Various aspects of the simulation (framerate, grid overlay, etc.) may be edited to allow the user with different visual aspects of the running simulation.

References:

[1] Berto, Francesco and Jacopo Tagliabue, "Cellular Automata", The Stanford Encyclopedia of Philosophy (Fall 2017 Edition), Edward N. Zalta (ed.) [Online]. Available: https://plato.stanford.edu/archives/fall2017/entries/cellular-automata/. [Accessed : Nov. 3, 2020]

[1] V. J. Blue, M. J. Embrechts and J. L. Adler, "Cellular automata modeling of pedestrian movements," 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, Orlando, FL, USA, 1997, pp. 2320-2323 vol.3, doi: 10.1109/ICSMC.1997.635272.