|  |
| --- |
| Carleton University |
| Bi-directional Pedestrian Walkway Modeling and Simulation with CELL-DEVS |
| SYSC5104 |

|  |
| --- |
| Wang Xiang 100680824  11/20/2010 |

# Introduction:

In this assignment, a published paper with a title of “An Improved Cellular Automation Model for Urban Walkway Bi-directional Pedestrian Flow” [1] is studied and implemented based on the suggested model and rules. The original work tends to study the behaviours of pedestrians with different walking speed and different actions based on some probabilities. There are 8 cellular automata rules the article suggested as listed below:

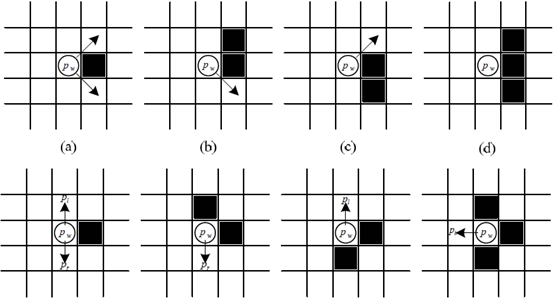


Figure : Cell-DEVS rules for pedestrians

These 8 rules can be further split into more rules because each black square can represent an obstacle or a pedestrian of either direction. With the provided 8 rules and their expansions, there are a huge amount of different cases to be considered for the CELL-DEVS implementation. Thus the implementation of this assignment only uses a few of the rules with some modification. Please see next section for the conceptual model.

# Project Definition and Conceptual Design:

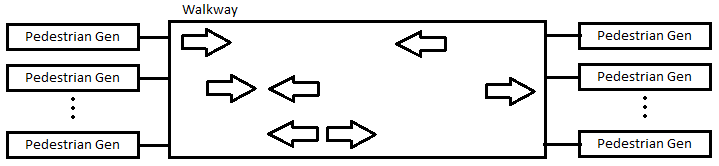
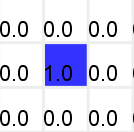
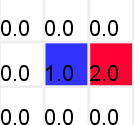


Figure : Cell-DEVS coupled model for the project

Figure 2 shows the coupled model definition of the project. The “Pedestrian Gen” blocks are CD++ built in random sequence generators. The “Walkway” is a Cell-DEVS model of size (8\*20) with the top most and the bottom most rows represent the walls of the walkway. Pedestrians are generated at the left and right ends of the walkway. The goal of the project is to program the “pedestrians” to travel to the opposite end of the walkway without collision. The speeds of all “pedestrians” are modeled at a constant rate, and everyone travels at that same speed. To do this, some rules are used to regulate the behaviours of the pedestrians as explain below in detail:

**Rule 1: Moving forward rule.**

If the cell in front of a “pedestrian” from either direction is empty, and no other “pedestrian” from the opposite direction is located two cells ahead, move forward. This is the highest priority rule. No action from other rules can take the empty square of moving forward rule.

**Rule 2: Side stepping when straight movement is blocked.**

If a “pedestrian” is blocked by the opposite side “pedestrian”, both “pedestrians” try to step to the right if that cell is empty. If the “pedestrian” is blocked by another one traveling to the same direction, simply do nothing but wait the other one to move first. This is the second highest priority rule.

**Rule 3: Side stepping when trying to take the same cell for straight movement.**

If two “pedestrians” from opposite side trying to take the same empty cell, both step to the right. If they are travelling in the same direction, Rule 1 takes care of it. This is the third highest priority rule.

**Rule 4: Side step to the left if the right side is a wall.**

For the cases where side step to the right is impossible because they are by a wall, they the “pedestrians” try to step to the left if there is an empty cell. This rule applies to both side-stepping when blocked or trying to take the same cell.

These rules are subset or a generalization of the 8 rules provided in the literature [1] according to human natural way to walking. Moving straight forward is the most fundamental and most important form of traveling. Pedestrians change lane only when necessary. Also, people tend to step to the right when encounter another pedestrian traveling the opposite direction. Given everyone travel at same speed, it does not make sense to make a side step when encounter someone with the same direction as the pedestrian.

Having the project definition and rules of Cell-DEVS model understood, the next section gives a formal specification of the project.

# CELL-DEVS Formal Specification:

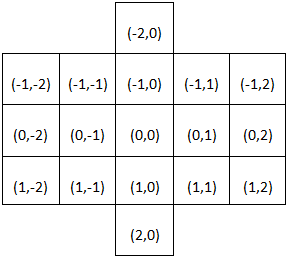


Figure : Neighbourhood definition

**M = <Xlist, Ylist, I, X, Y, ƞ, N, {r,c}, C, B, Z, select>**

**Xlist = {(-1,-2)…(1,-2) (-1,2)…(1,2)};**

**Ylist = {};**

**I = <Px, Py>; with Px = {<X(-1,-2), binary>…<X(1,2),binary>}; Py={};**

**N = {(-2,0), (-1,-2), (-1,-1), (-1,0), (-1,1), (-1,2), (0,-2), (0,-1), (0,0), (0,1), (0,2), (1,-2), (1,-1), (1,0), (1,1), (1,2), (2,0)};**

**Ƞ = 17;**

**r = 8; c = 20;**

**X=Y={0, 1, 2}:**

**B = not-wrapped**

**C = {Cij | i ϵ [1, 6], j ϵ [0, 19]};**

**Z: Inverse neighbourhood of N**

**Select = {(0,1), (0,-1), (-1,0), (1,0)};**

# Project Implementation

This section will present the pseudo code implementation of the rules of the cellular automata. Also the expected behaviours of the rules will be explained.

**Entrance rule:**

Assign a value of 2 to the pedestrians who are walking towards left if the right most cells receive external inputs, and their own values are either 0 or 1.

Assign a value of 1 to the pedestrians who are walking towards right if the left most cells receive external inputs, and their own values are either 0 or 2.

**Exit rule:**

If cells with value of 1 reach the left most cells, exit.

If cells with value of 2 reach the right most cells, exit.

**Straight movement rule:**

Move forward (left or right) if the cell in front of the current cell is empty, and two cells ahead does not have a cell moving in the opposite direction.

**Side step rule when a cell is blocked:**

For cells with value of 1, move to the cell below; for cells with value of 2, move to the cell above. Perform the moving action only when the cells they attempt to go to is empty, and there is no straight movement into the cell.

**Side step rule when two cells attempts to occupy the same cell as a result of straight movement:**

Same strategy as the “Side step rule when a cell is blocked” rule, but with different neighbours to check.

**Rule for boarder behaviour:**

If cells with value of 2 attempt to step to the cell above and found out the above row is the “wall”, the cell then try to move to the cell below. The opposite rule applies to the cells with value of 1.

A careful reader would notice that the above rule set does not check for possible collision due to various side stepping. The reason of this is mainly due to the complexity of the program. If taking into the consideration of side stepping, then for every rule there is a need to check the rest of all possible collisions with two directions of travelling. The neighbourhood in this case will become a 6\*7 space, which almost equivalent to half of defined model area. It also does not make sense in reality that a pedestrian attempts to change lane needs to check half of all peoples on that path. Most importantly, a possible solution will be implemented in an update of this version with introducing of another plane.

# Testing of the project implementation

This section will present the testing results of the project. The Cell-DEVS animation tool will be used to demonstrate the results. A simulation video can be found at the following link:

http://www.youtube.com/watch?v=6xvDVrhgNMM

**Entrance rule testing:**

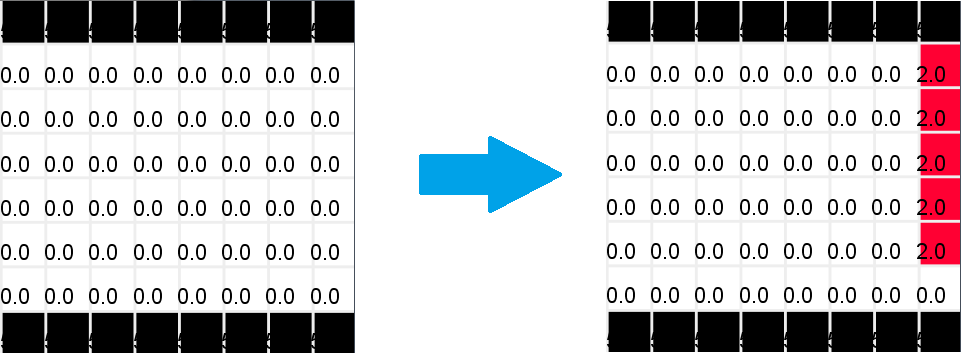


Figure : Entrance rule testing 1

The most basic entrance rule, when there is no one on the edge of the walkway, enter freely.

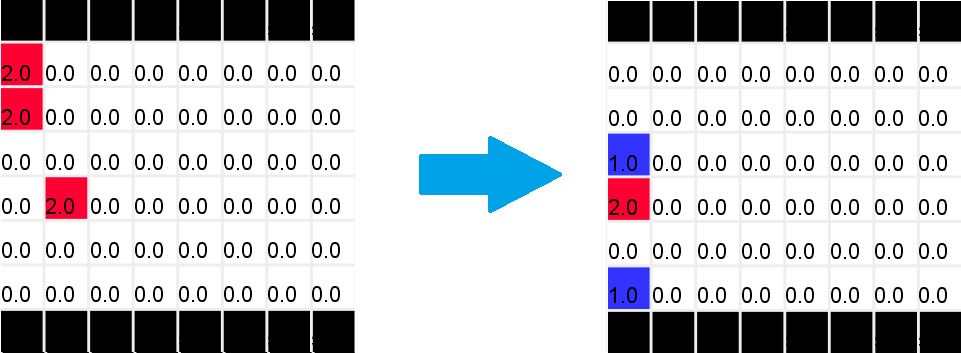


Figure : Entrance rule testing 2

While the pedestrian from the right (two cells with value of 2) leaves the modeled space, new entrance from left (cells with value of 1) enters the space.

As shown in the course of simulation, entrance behaves as expected, no collision happens, nor any sight of losing of a cell.

**Exit rule testing:**

Please refer to Figure 5 to see the illustration of the cells leaving the modeled space. Over the course of the simulation, the behaviour of the rule functions well.

**Straight movement rule testing:**

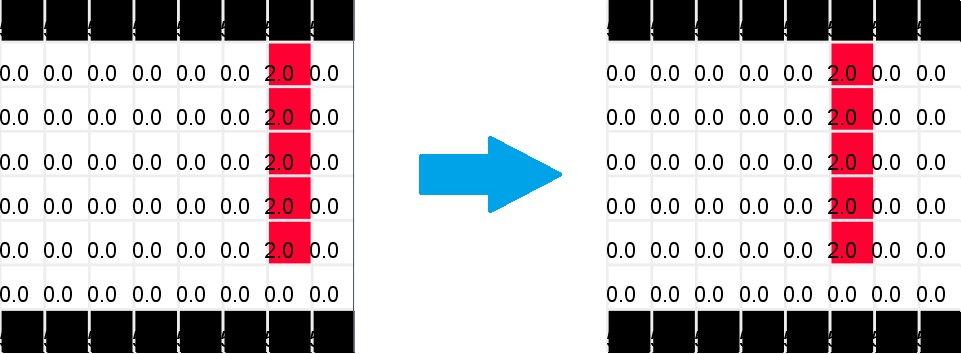


Figure : Straight movement rule testing 1

For each of the 5 cells traveling toward left, since there is nothing in front of them they move forward together.

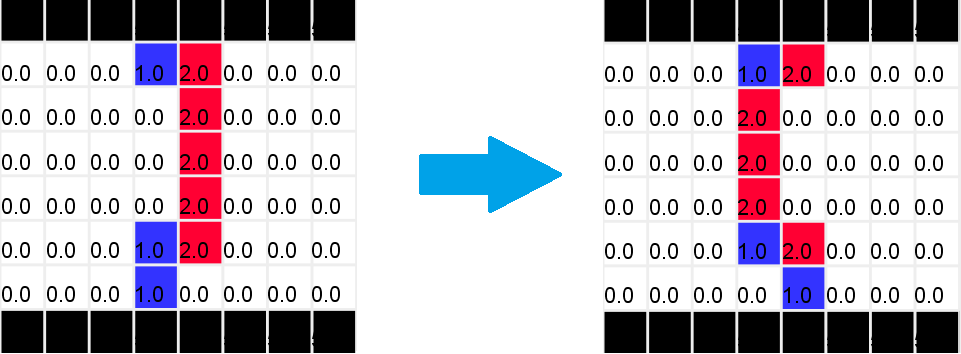


Figure : Straight movement rule testing 2

Cells from both directions travel forward together. The blocked ones dose not perform side step in this case because there is no empty space above or below, or there is a cell moving straight into the empty cell. Overall, straight movement behaves as expected and no collision observed.

**Side stepping when block rule testing:**

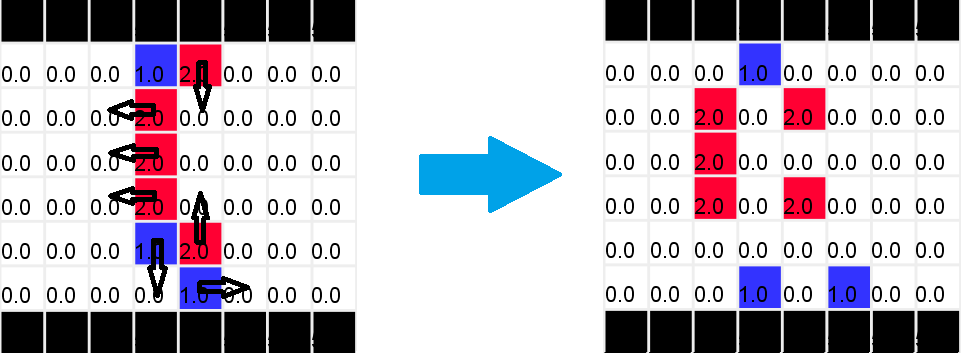


Figure : Side stepping rule when blocked testing 1

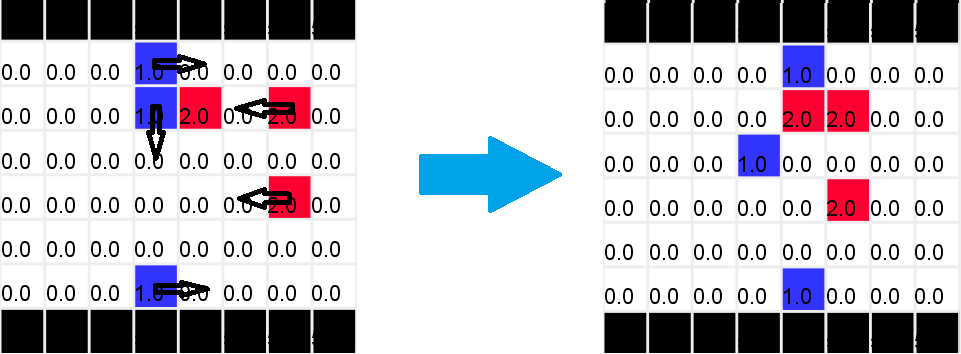


Figure : Side stepping rule when blocked testing 2

As illustrated with the arrows, the side stepping rule behaves well. Over the course of this particular simulation, there is no collision detected due to this side stepping. However it is seen in other simulations. The result is a losing of a cell which analogy to losing a pedestrian upon collision. This is not very good. Therefore, the next version of this implementation can hopefully solve this problem.

**Side stepping due to heading to same cell testing:**

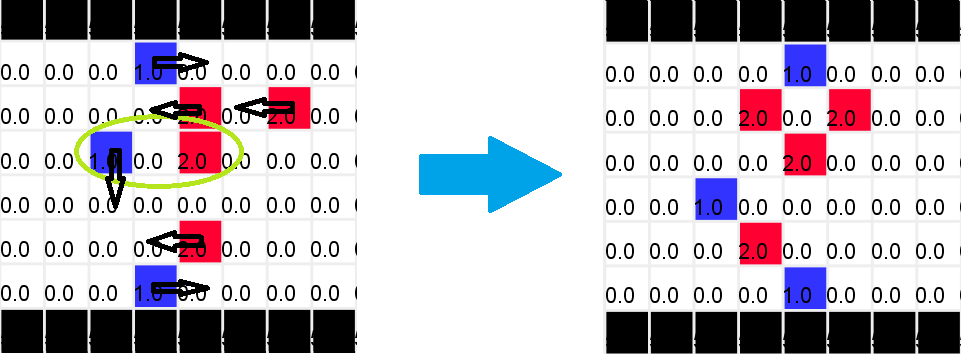


Figure : Side stepping due to heading to same cell testing

Same as the “side stepping when blocked” rule, over the course of this simulation, the “side stepping due to heading to same cell” rule behaves very well too. No collision is detected.

**Boarder behaviour rule testing:**

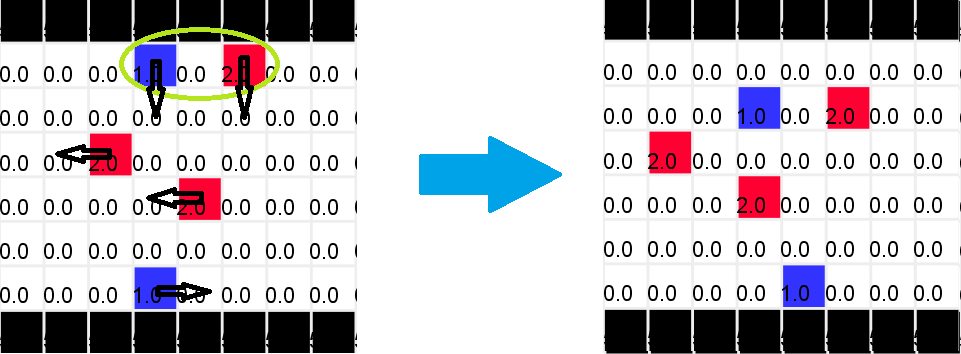


Figure : boarder behaviour rule testing 1

As shown with the green circle, two pedestrians attempts to move into the same cell and thus need to perform side step. Originally, they both should step to the right of their walking direction, but the red cell is not able to because there is a wall at its right, so it stepped to the left.

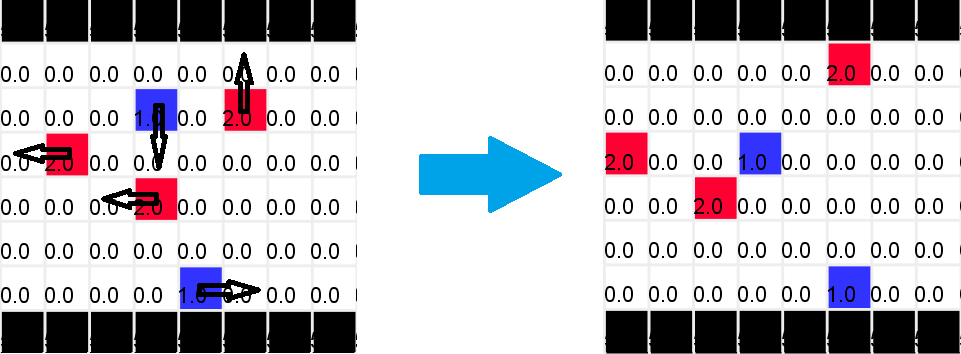


Figure : Boarder behaviour rules testing 2

After the movement illustrated in Figure 11, this figure shows the next action of the cells.

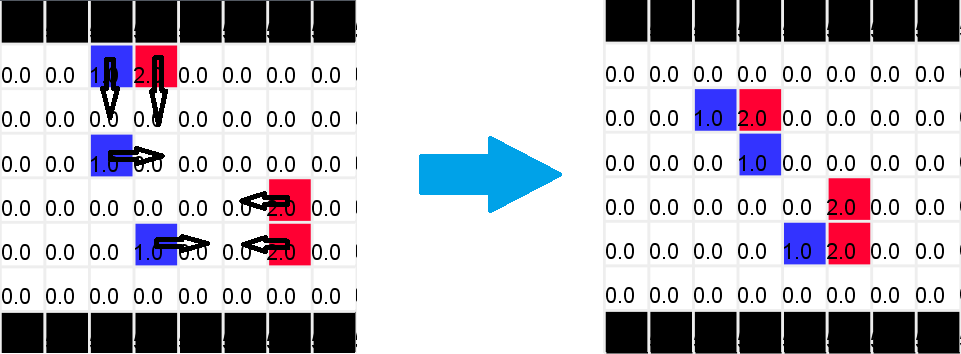


Figure : Boarder behaviour rules testing 3

Figure 11 shows the boarder behaviour with two cells heading to the same cell. Figure 13 shows the boarder behaviour when blocked.

# Conclusion:

To conclude, this rule set behaves very well. For all simulations performed, there was no traffic gem observed. Pedestrians flow very naturally. In this particular simulation, there is no collision detected at all. However, since the verification of possible collision due to side stepping is not implemented in this project, there are certain rare cases where two cells merge into one observed in some other simulations.

One proposed way to fix the problem is to assign priority to the rules, and introduce another plane that records which rule has already been applied to the cells. Hopefully this can prevent multiple rules been used on the same cell, and hopefully can simplify the implementation.

# Reference:

[1] T. Wang, J. Chen. An Improved Cellular Automation Model for Urban Walkway Bi-directional Pedestrian Flow: International Conference on Measuring Technology and Mechatronics Automation, 2009.