Methodologies for Discrete Event Modelling and Simulation

Carleton University

March 24, 2025

Belal Alobieda 101151327

Random-Walk Ants Project

Introduction

The purpose of this project is to demonstrate an understanding of modeling a system using the Cell-DEVS formalism and implementing it in the Cadmium simulator. This time, we simulate a random-walk ants scenario: each cell on a 2D grid can be either empty or occupied by an ant facing one of four directions (north, east, south, west). At each time step, ants try to move forward if the target cell is free; otherwise, they choose a new random direction. This approach showcases how local rules in a Cell-DEVS model can yield interesting emergent behaviors on a larger grid.

CELL DEVS Models

Atomic Models : walkcell

This atomic Cell-DEVS model represents a single cell that can either be empty or have an occupant. The occupant has two properties:

- direction: A value can be (0,1,2,3,4), where 0 means the cell is empty, 1 means the occupant is facing north, 2 is east, 3 is south, and 4 is west.
- randomVal: A numeric value used as a priority, to decide which occupant takes the cell if multiple try to move into it.

At each time step, the cell behaves as follows:

- If the cell is empty (direction = 0), it looks at its neighboring cells. If one or more neighbors are facing the empty cell, the one with the highest randomVal will move in.
- If the cell is occupied, the occupant tries to move in the direction it is currently facing. If the destination cell is empty, the occupant moves there. If the destination cell is already taken, the occupant picks a new direction and possibly a new randomVal.

1. Formal Specification: randomWalkCell

CD = ⟨X, Y, I, S, θ, N, d, δint, δext, τ, λ, D⟩

- **X = Y = ∅**
- **S** =
  { direction ∈ {0, 1, 2, 3, 4},
  randomVal ∈ ℤ
  }
- **N** = {(-1,0), (1,0), (0,-1), (0,1)}
- **delay** = transport delay
- **d** = 1.0
- **δint**:
- **δext**: τ
- **τ** = N → S:
  1. If direction = 0 (empty), choose occupant from neighbors facing this cell with highest randomVal to move in.
  2. If direction ≠ 0 (occupied), attempt to move to the cell in direction; if blocked, pick a new direction randomly.
- **λ** = current state
- **D** = 1.0

Coupled Models : **randomWalkGrid**

CCA = ⟨S, n, C, η, N, T, τ⟩

- **S** = same state set as the atomic model { direction, randomVal }
- **n** = 2
- **C** = C { (i,j) | i,j ∈ [0 (width-1)] }
- **η** = 1
- **N** = { (dx,dy) | dx,dy ∈ {-1,0,1} }
- **Border** = wrapped

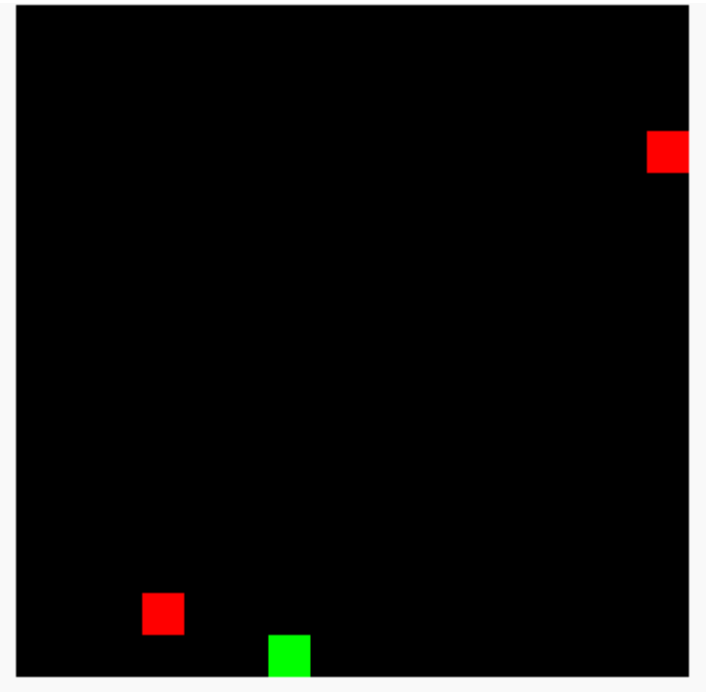
- **Rules** :

```
IF current_direction = 0 THEN
  next_direction ← find_neighbor_with_highest_randomVal_facing cell()
ELSE
  IF target_cell_is_empty() THEN
    current_cell_becomes_empty()
  ELSE
    current_direction ← pick_new_random_direction()
  ENDIF
ENDIF
```

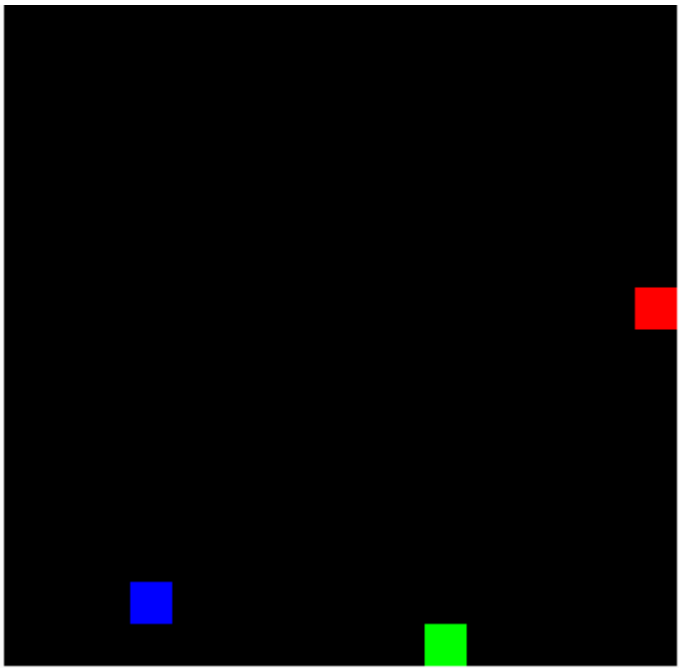
Simulation Results

```
28;64;(6,15);;<3,4>
28;75;(1,15);;<0,0>
28;78;(15,7);;<0,0>
28;80;(0,4);;<0,0>
28;114;(5,0);;<0,0>
28;116;(4,0);;<0,0>
28;120;(3,0);;<0,0>
28;121;(5,14);;<0,0>
28;123;(2,0);;<0,0>
28;135;(7,15);;<0,0>
28;157;(4,1);;<0,0>
28;158;(15,4);;<4,7>
28;159;(1,6);;<0,0>
28;161;(3,1);;<0,0>
28;163;(3,13);;<4,7>
28;175;(13,5);;<0,0>
28;181;(5,15);outputNeighborhood;<3,4>
28;181;(5,15);;<0,0>
28;186;(15,3);;<0,0>
28;190;(3,12);;<0,0>
28;195;(14,7);;<0,0>
28;196;(14,4);;<0,0>
28;198;(0,5);;<0,0>
28;201;(6,0);;<0,0>
28;211;(15,8);;<0,0>
28;212;(0,7);;<0,0>
28;216;(1,5);;<0,0>
28;227;(14,5);;<0,0>
28;229;(15,5);outputNeighborhood;<4,7>
```

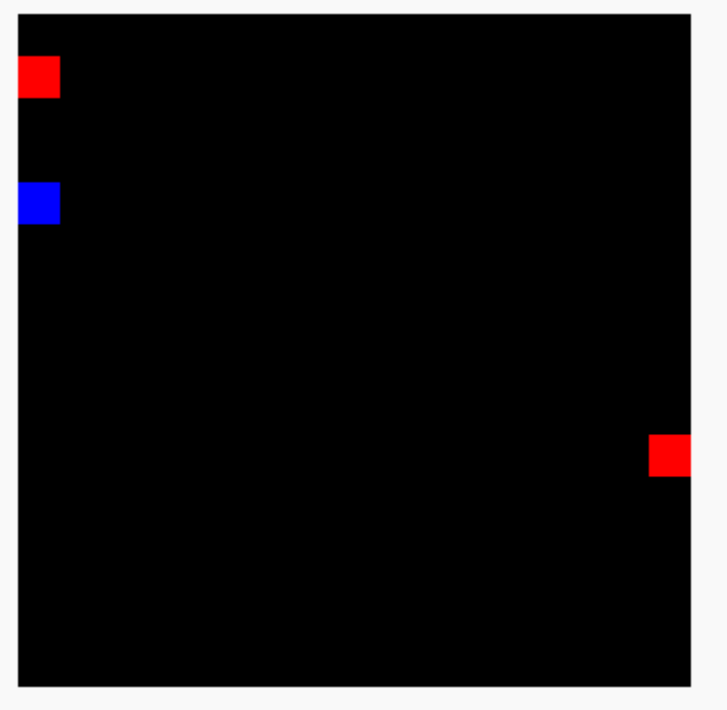
This screenshot shows a simulation log at time step 28, listing each cell update.
Each line includes the cell coordinates (6,15) and the new state <3,4>.
“outputNeighborhood” indicates a cell influencing its neighbors. However to help visulixing that to understand it much better. I used DEVS Simulation Viwer to visualize the changes. Below some screenshots for our understanding.



In this screenshot, the squares represent ants are trying to move. The colour represent the direction. So here we have two ants are moving downward the screen. While the green ant moving to the right of the screen.



In the screenshot, as we can see we have three simulated ants moving into three different directions. Red is downward, green is right, blue is upward .



In this screeshot, we have two simulated ants moving opposite directions. They both reach a grid they want to accupy. And the one who has higher randomVal will win and take the grid

In the screenshot, two simulated ants are moving right following each other to show that when the ant leave it is place(grid), it will be free for other ants to occupy it.

Additional tests