Discrete-Event Modeling Using Cell-DEVS

Assignment 2

SYSC 5104

Football Player Interaction Model (FPI)

Youssef Elfaramawy (101366514)

Department of Systems and Computer Engineering

Carleton University

March 2025

CONTENTS

PART I - CONCEPTUAL MODEL DESCRIPTION

Background

This model represents a simplified football match where a player's performance is influenced by physical fatigue and mental state. Players continually adjust their behavior on the pitch – such as moving into open space, passing, or dribbling – based on their current physical and mental conditions.

This model is inspired by Makarenko et al.'s "Towards Cellular Automata Football Models with Mentality Accounting. [1]" Their work simulates a football match using Cellular Automata (CA), considering two teams where each player's movement is determined probabilistically within a von Neumann neighborhood [1]. Player interaction is governed by a set of rules that dictate movement and ball possession [1].

While my model shares similarities in decision-based movement and mentality tracking [1], it differs in key areas. Unlike their probabilistic approach [1], my model explicitly regulates actions based on meeting defined mental and fatigue thresholds rather than weighted probabilities. Additionally, my model is limited to vertical directions, while theirs supports both horizontal and vertical movement [1].

A key challenge presented in their report on CA-based football models is the element of anticipation [1]. Anticipation, as defined in their report, is the ability of a player to predict the future state of the game [1]. The authors describe this aspect as mentality accounting [1]. In other words, players, at the atomic level, must decide their best course of action that will result in a higher chance of success (winning) [1]. However, this raises the need for complex mathematical models as well as internal state tracking [1]. Using Cell-DEVS, my model naturally addresses this challenge as each cell manages its own state. As for complex anticipation models, I introduced action costs instead to simulate mental and physical strain that could occur at any point in the game when players perform different actions. This helps maintain the modular nature of Cell-DEVS for discrete-event models while also incorporating the report's mentality accounting aspect.

Unlike my previous work, the DEVS Player Substitution System Model, the focus is shifted to individual players within a single team that retains possession of the ball throughout the simulation.

Model Overview

System Structure

The football pitch is represented as a discrete 2D grid, reflecting a common football formation. Rather than explicitly modeling complex ball physics, the ball follows player decisions (e.g. dribbling, passing) and movement. This decision-making is governed by a player's fatigue and mental state.

Sketch of Model Structure

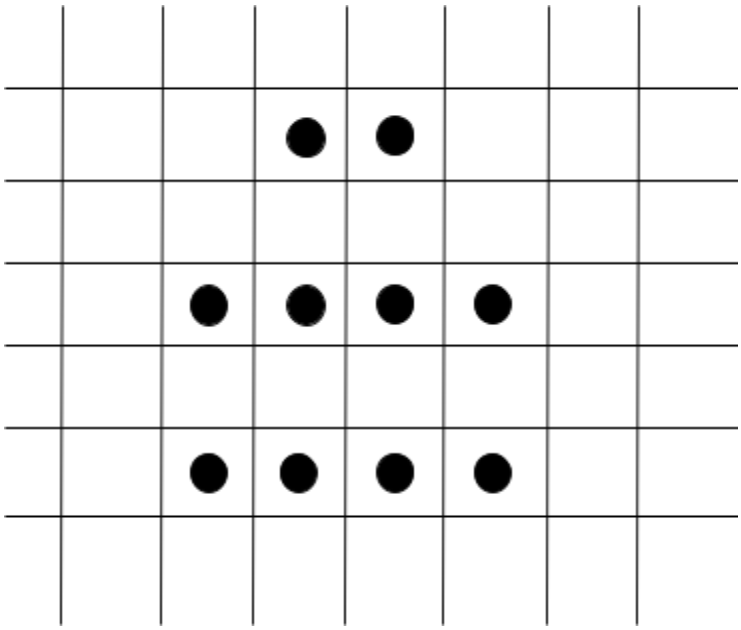

Figure 1: Conceptual Model of Football Pitch Grid

System Behavior

Player Behavior: Each occupied cell updates its state based on the following conditions:

- **Fatigue:**
    - Increases during actions like dribbling or passing.
    - Decreases over time when idle (no actions, no ball).
- **Mental:**
    - Decreases during actions like dribbling or passing.
    - Recovers slowly when idle (no actions, no ball)

Actions Selection Rules

- **In Possession of the Ball:**
    - **Low fatigue & High mental:** Dribbles forward/backward (moves up/down one cell) if space is available
    - **High fatigue & High mental:** Passes the ball (long pass) to a teammate in the next zone if the teammate is available. Otherwise, can pass the ball to a zone behind
    - **High fatigue & Low mental:** Passes the ball (short pass) to a teammate within the same line (east or west)
- **Not in Possession of the Ball:**
    - A player moves up/down one cell if:
        - A teammate in the same line (east or west) dribbles forward/backward
        - Their fatigue is below a threshold
        - Their mental is above a threshold
        - The adjacent cell is unoccupied (no player)

Ball Interaction

- **Dribbling:** When a player moves, the ball moves with them
- **Passing:** When a pass occurs, the ball moves to the target player's cell

Cell Configuration Parameters

Each cell tracks the following state variables:

- **has_player (bool):** Indicates whether a player occupies a cell
- **has_ball (bool):** Indicates whether the player in the cell possesses the ball
- **mental (double):** Represents the player's mental state
- **fatigue (double):** Represents the player's fatigue level
- **action (string):** Represents the player's action (dribble, move, short pass, long pass, hold)
- **direction (string):** Represents the direction of dribble, move, pass
- **inactive_time (int):** Represents how long an unoccupied cell has remained idle

Delay Type

Transport delay (d = 1.0) allows for event propagation in the system by not preempting state changes.

Assumptions

- The ball is always possessed by a player and cannot exist in an empty cell.
- Player movement is constrained only by occupied cells, not by predefined zones (defense, midfield, attack).
- Zone-based movement restrictions (e.g., midfielders not dribbling into attack) are not enforced but may be introduced later.

PART II - IMPLEMENTATION

Formal Specification

Football Player Atomic Cell Model <X, Y, S, N, d, τ, δ_{int}, δ_{ext}, λ, t_a>

X = ∅

Y = ∅

S = { has_player ∈ {true, false}, has_ball ∈ {true, false}, mental ∈ [0, 100], fatigue ∈ [0, 100], action ∈ {None, short_pass, long_pass, dribble, move, hold}, direction ∈ {None, north, south, east, west}, inactive_time ∈ ℕ }

N = { (-2,0), (-1,0), (0,0), (1,0), (2,0), *(Extended von Neumann Neighborhood)*
(0,-2), (0,-1), (0,1), (0,2),
(-1,-1), (-1,1), (1,-1), (1,1) }

d = 1 time unit (*Transport Delay)*

τ: N → S (Local Computation Rules)

The local computation rules define how a player's states update based on neighborhood inputs and/or interactions.

1. **Receiving a pass**
    - If a player receives a *short pass* from the east/west (direct neighbors), they gain possession of the ball, and mental/fatigue levels are adjusted accordingly.
        - S = { has_player = true, has_ball = true, mental -= 1.0, fatigue += 2.5}
    - If a player receives a *long pass* from the north/south (direct and extended neighbors), they gain possession of the ball, and mental/fatigue levels are adjusted accordingly.
        - S = { has_player = true, has_ball = true, mental -= 2.0, fatigue += 3.5}

2. **Making a pass**
    - If 70 > fatigue > 30 and mental < 60, the player makes a *short pass* to a nearby teammate (east or west).
        - S = { has_player = true, has_ball = false, action = short_pass, direction ∈ {east, west}, mental -= 1.5, fatigue += 3.0}
    - If fatigue > 25 and 70 > mental > 60, the player makes a *long pass* to a teammate 1 or 2 steps ahead (north or south).
        - S = { has_player = true, has_ball = false, action = long_pass, direction ∈ {north, south}, mental -= 1.5, fatigue += 3.0}
3. **Dribbling**
    - If fatigue < 50 and mental ≥ 70, the player dribbles forward (north/south), moving into an open space.
        - S = { has_player = true, has_ball = true, action = dribble, direction ∈ {north, south}, mental -= 3.0, fatigue += 7.0}
4. **Moving without the Ball**
    - If fatigue < 40 and mental > 50 and a direct neighbor (east/west) dribbles north or south, the player moves forward or backward with him into an open space.
        - S = { has_player = true, has_ball = false, action = move, direction ∈ {north, south}, mental -= 2.0, fatigue += 5.0}
5. **Holding the Ball**
    - If a player can't perform a dribble or pass action, he/she holds the ball, reducing mental and fatigue levels over time.
        - S = { has_player = true, has_ball = true, action = hold, mental -= 1.0, fatigue += 2.0}
6. **Mental & Fatigue Recovery**
    - When no actions are performed, player mental and fatigue levels slowly rise over time.
        - S = { has_player = true, has_ball = false, action = None, direction = None, mental += 1.0, fatigue -= 1.0}

δ_{int}, δ_{ext}, λ, t_a are defined using Cell-DEVS specifications.

Phase-Based Event Flow in the Cell-DEVS Model

The execution of the Football Player Cell-DEVS follows a structured phase-based event flow, ensuring that player actions/interactions are processed and propagated correctly over time. This approach divides the simulation cycle (single time step) for each cell to operate as follows: data collection, action decision, receiving delayed actions, and cleanup. Each phase is crucial to maintaining a consistent sequential update of states, allowing for a synchronized simulation.

1. Data Collection

At the beginning of each simulation cycle, a cell gathers information from its neighboring cells (*extended von Neumann*) to track different states, allowing it to assess its available options. This includes:

- The presence of teammates in adjacent or extended neighborhood cells.
- Whether a neighboring cell has recently performed a *dribble* or a *pass*.
- The current mental and fatigue state of a player looking to *move* or *dribble*, for state attribute inheritance.

2. Action Decision

Once the data is collected, a cell evaluates it and its current conditions (mental and fatigue levels) to determine whether an action can be taken. The decision-making process follows a hierarchical structure:

- If a player has possession of the ball, he/she can attempt to pass, dribble, move, or hold based on predefined thresholds and neighboring conditions.
- If a player does not have possession of the ball, he/she can attempt to move if a neighboring player dribbles and they pass predefined thresholds. They can also choose to remain in place and recover.

Constraints are placed on mental and fatigue thresholds to try to prevent overlapping conditions. In the event a player satisfies more than one condition (rule), priority is given to pass over dribble.

3. Receiving Delayed Actions

In this phase, delayed neighborhood inputs (state transitions are broadcasted after d = 1.0) are processed. Actions such as passing, dribbling, and moving from the previous cycle are finalized in this phase. This allows for correct propagation when players pass or move/dribble into a target cell. This phase is responsible for:

- Transferring the ball to a target player, when a *short* or *long pass* occurs.
- Moving a player to a target cell, when an action of *dribble* or *move* occurs. This also includes the target cell inheriting the source cell's mental/fatigue attributes.

Note: Due to the modular nature of Cell-DEVS, each cell can only modify its state and not that of others. Consider the following example:

- If Cell (2, 2) wants to dribble north to Cell (1, 2), an empty cell, the following needs to occur:
    - Cell (2,2) sends an action to dribble, causing it to lose player and ball.
        - S = { has_player = true, has_ball = true, action = None, direction = None, mental = 70, fatigue = 30}
        - S = { has_player = false, has_ball = false, action = dribble, direction = north, mental = 67, fatigue = 37}
    - Cell (1,2) sees action *dribble*, inherits Cell (2,2) mental/fatigue attributes, and gains a player and a ball.
        - S = { has_player = false, has_ball = false, action = None, direction = None, mental = 50 (default), fatigue = 0 (default)}
        - S = { has_player = true, has_ball = true, action = None, direction = None, mental = 67, fatigue = 37}

4. Cleanup

This final phase ensures that cells that have moved/dribbled during the previous step are correctly reset after a certain time. This includes:

- Maintaining source cell output for one cell, ensuring their actions are correctly detected by neighboring cells.
- Clearing *action* and *direction* state attributes to prevent redundant updates in future cycles. This ensures that the next cycle starts with a more accurate presentation of the current pitch (grid) state.

This cleanup step helps prevent constant state updates that carry over during the simulation cycle.

Testing Strategies and Documentation

Given the complexity of player and ball interactions, a hierarchically structured testing approach was adopted. A smaller 3x3 grid framework was used to reduce the number of players and isolate the testing scope to a specific action (*dribble*, *short pass*, *long pass*),. This allowed for a testing environment that is both controlled and observable when looking at the outputs in the CSV log file.

Each of the following test cases, as mentioned, will focus on a specific gameplay aspect before transitioning to a wider experimental frame.

Test Case #1: Short Passing Between Two Players (3x3)

Description

This test aimed to validate that players could successfully perform short pass actions to one another. The test was configured as follows:

- Three Players positioned in adjacent cells
- Middle Cell starts with the ball
- Middle Cell and Right Cell meet the criteria for a **short pass** (fatigue and mental meet thresholds)

Results

```
time;model_id;model_name;port_name;data
0;5;(2,0);;{has_player: 1, has_ball: 0, mental: 50, fatigue: 0, action:
None, direction: None, inactive_time: 0}
0;7;(2,1);;{has_player: 1, has_ball: 1, mental: 30, fatigue: 35, action:
None, direction: None, inactive_time: 0}
0;9;(2,2);;{has_player: 1, has_ball: 0, mental: 30, fatigue: 35, action:
None, direction: None, inactive_time: 0}
0;7;(2,1);outputNeighborhood;{has_player: 1, has_ball: 1, mental: 30,
fatigue: 35, action: None, direction: None, inactive_time: 0}
0;7;(2,1);;{has_player: 1, has_ball: 0, mental: 28.5, fatigue: 38, action:
short_pass, direction: east, inactive_time: 0}
1;9;(2,2);;{has_player: 1, has_ball: 1, mental: 31, fatigue: 35.5, action:
None, direction: None, inactive_time: 0}
2;9;(2,2);outputNeighborhood;{has_player: 1, has_ball: 1, mental: 31,
fatigue: 35.5, action: None, direction: None, inactive_time: 0}
2;9;(2,2);;{has_player: 1, has_ball: 0, mental: 29.5, fatigue: 38.5,
action: short_pass, direction: west, inactive_time: 0}
```

```
3;7;(2,1);;{has_player: 1, has_ball: 1, mental: 30.5, fatigue: 37.5,
action: None, direction: None, inactive_time: 0}
```

The output CSV file confirms that:

- Player (2,1) loses possession of the ball after short passing east.
- Player (2,2) gains possession of the ball in the next cycle.
- Player (2,2) loses possession of the ball after short passing west.
- Player (2,1) gains back possession of the ball in the next cycle.

Discussion

Looking at the CSV file above, we can observe that short passes were correctly executed. Additionally, the ball transfer delay was observed as intended, and the target player ended up receiving the ball. This confirms:

- The short pass rule is valid and action costs are correctly implemented.
- The source player loses the ball when passing.
- The target player receives the ball after a delay (d = 1).

Test Case #2: Dribbling and Neighbor Reaction (3x3)

Description

This test aimed to validate that players could successfully dribble forward/backward, and if a neighboring cell (east/west) detected that action, it moved accordingly. The test was configured as follows:

- Three Players positioned in adjacent cells
- Middle Cell starts with the ball
- Middle Cell short passes to Right Cell
- Right Cell meets the criteria for **dribble** (fatigue and mental meet thresholds)
- Middle Cell meets the criteria for **move** (fatigue and mental meet thresholds)

Results

```
time;model_id;model_name;port_name;data
0;5;(2,0);;{has_player: 1, has_ball: 0, mental: 50, fatigue: 0, action:
None, direction: None, inactive_time: 0}
0;7;(2,1);;{has_player: 1, has_ball: 1, mental: 55, fatigue: 35, action:
None, direction: None, inactive_time: 0}
0;9;(2,2);;{has_player: 1, has_ball: 0, mental: 80, fatigue: 10, action:
None, direction: None, inactive_time: 0}
```

```
0;7;(2,1);outputNeighborhood;{has_player: 1, has_ball: 1, mental: 55,
fatigue: 35, action: None, direction: None, inactive_time: 0}
0;7;(2,1);;{has_player: 1, has_ball: 0, mental: 53.5, fatigue: 38, action:
short_pass, direction: east, inactive_time: 0}
1;9;(2,2);;{has_player: 1, has_ball: 1, mental: 79, fatigue: 12.5, action:
None, direction: None, inactive_time: 0}
2;7;(2,1);;{has_player: 1, has_ball: 0, mental: 53.5, fatigue: 38, action:
short_pass, direction: east, inactive_time: 0}
2;9;(2,2);outputNeighborhood;{has_player: 1, has_ball: 1, mental: 79,
fatigue: 12.5, action: None, direction: None, inactive_time: 0}
2;9;(2,2);;{has_player: 0, has_ball: 0, mental: 79, fatigue: 12.5, action:
dribble, direction: north, inactive_time: 1}
3;7;(2,1);;{has_player: 0, has_ball: 0, mental: 53.5, fatigue: 38, action:
move, direction: north, inactive_time: 1}
3;9;(2,2);;{has_player: 0, has_ball: 0, mental: 50, fatigue: 0, action:
None, direction: None, inactive_time: 0}
4;1;(1,2);outputNeighborhood;{has_player: 1, has_ball: 1, mental: 76,
fatigue: 19.5, action: None, direction: None, inactive_time: 0}
4;1;(1,2);;{has_player: 0, has_ball: 0, mental: 76, fatigue: 19.5, action:
dribble, direction: north, inactive_time: 1}
4;6;(1,1);;{has_player: 1, has_ball: 0, mental: 51.5, fatigue: 43, action:
None, direction: None, inactive_time: 0}
4;7;(2,1);;{has_player: 0, has_ball: 0, mental: 50, fatigue: 0, action:
None, direction: None, inactive_time: 0}
5;1;(1,2);;{has_player: 0, has_ball: 0, mental: 50, fatigue: 0, action:
None, direction: None, inactive_time: 0}
5;2;(0,2);;{has_player: 1, has_ball: 1, mental: 73, fatigue: 26.5, action:
None, direction: None, inactive_time: 0}
5;5;(2,0);;{has_player: 1, has_ball: 0, mental: 51, fatigue: 0, action:
None, direction: None, inactive_time: 0}
5;6;(1,1);outputNeighborhood;{has_player: 1, has_ball: 0, mental: 51.5,
fatigue: 43, action: None, direction: None, inactive_time: 0}
5;6;(1,1);;{has_player: 1, has_ball: 0, mental: 52.5, fatigue: 42, action:
None, direction: None, inactive_time: 0}
5;7;(2,1);outputNeighborhood;{has_player: 0, has_ball: 0, mental: 50,
fatigue: 0, action: None, direction: None, inactive_time: 0}
5;7;(2,1);;{has_player: 0, has_ball: 0, mental: 50, fatigue: 0, action:
None, direction: None, inactive_time: 0}
5;8;(1,0);;{has_player: 0, has_ball: 0, mental: 50, fatigue: 0, action:
None, direction: None, inactive_time: 0}
```

```
5;9;(2,2);;{has_player: 0, has_ball: 0, mental: 50, fatigue: 0, action:
None, direction: None, inactive_time: 0}
6;2;(0,2);;{has_player: 0, has_ball: 0, mental: 73, fatigue: 26.5, action:
dribble, direction: south, inactive_time: 1}
6;6;(1,1);;{has_player: 1, has_ball: 0, mental: 53.5, fatigue: 41, action:
None, direction: None, inactive_time: 0}
7;1;(1,2);;{has_player: 1, has_ball: 1, mental: 70, fatigue: 33.5, action:
None, direction: None, inactive_time: 0}
7;2;(0,2);;{has_player: 0, has_ball: 0, mental: 50, fatigue: 0, action:
None, direction: None, inactive_time: 0}
```

The output CSV confirms that:

- Player (2,1) loses possession of the ball after short passing east.
- Player (2,2) gains possession of the ball in the next cycle.
- Player (2,2) dribbles north to cell (1, 2).
- Player (2,1) detects neighbor action and moves north to cell (1,1).
- Player (1,2) dribbles north to cell (0, 2).
- Player (2,1) remains in place as it doesn't meet move threshold
- Player (0,2) dribbles south to cell (1,2).

Discussion

Looking at the CSV file above, we can observe that the dribbling and moving actions were correctly executed. Additionally, the player and/or ball delay was observed as intended, and target cells ended up receiving them. This confirms that:

- The dribble rule is valid and action costs are correctly implemented.
- The source cell loses its player and ball when dribbling.
- The target empty cell receives the player and ball after a delay (d = 1).
- The move rule is valid and action costs are correctly implemented.
- The source cell loses its player when moving.
- The target empty cell receives the player after a delay (d = 1).

Test Case #3: Long Passing Between Two Players (3x3)

Description

This test aimed to validate that players could successfully perform long-pass actions to one another. The test was configured as follows:

- Two Players are positioned in the same column but in different rows (vertically aligned)

- Bottom Cell starts with the ball
- Bottom Cell and Top Cell meet the criteria for a **long pass** (fatigue and mental meet thresholds)

Results

```
time;model_id;model_name;port_name;data
0;8;(2,1);;{has_player: 1, has_ball: 1, mental: 65, fatigue: 30, action:
None, direction: None, inactive_time: 0}
0;9;(0,1);;{has_player: 1, has_ball: 0, mental: 65, fatigue: 30, action:
None, direction: None, inactive_time: 0}
ental: 50, fatigue: 0, action: None, direction: None, inactive_time: 0}
0;8;(2,1);;{has_player: 1, has_ball: 0, mental: 62.5, fatigue: 34, action:
long_pass, direction: north, inactive_time: 0}
0;9;(0,1);outputNeighborhood;{has_player: 1, has_ball: 0, mental: 65,
fatigue: 30, action: None, direction: None, inactive_time: 0}
1;8;(2,1);;{has_player: 1, has_ball: 0, mental: 62.5, fatigue: 34, action:
None, direction: None, inactive_time: 0}
2;9;(0,1);;{has_player: 1, has_ball: 0, mental: 60.5, fatigue: 37.5,
action: long_pass, direction: south, inactive_time: 0}
3;8;(2,1);;{has_player: 1, has_ball: 1, mental: 60.5, fatigue: 37.5,
action: None, direction: None, inactive_time: 0}
3;9;(0,1);;{has_player: 1, has_ball: 0, mental: 60.5, fatigue: 37.5,
action: None, direction: None, inactive_time: 0}
```

The output CSV confirms that:

- Player (2,1) loses possession of the ball after long passing north.
- Player (0,1) gains possession of the ball in the next cycle.
- Player (0,1) loses possession of the ball after long passing south.
- Player (2,1) gains possession of the ball in the next cycle.

Discussion

Looking at the CSV file above, we can observe that long passes were correctly executed. Additionally, the ball transfer delay was observed as intended, and the target player ended up receiving the ball. This confirms:

- The long pass rule is valid and action costs are correctly implemented.
- The source player loses the ball when passing.
- The target player receives the ball after a delay (d = 1).

Experimentation

For experimentation, the focus was on scaling up the simulation to a 10x10 grid with 10 players. The goal here was to produce a dynamic environment to examine whether the phase-based approach, discussed earlier, was robust and correctly propagated events.

Objective

The goal was to observe a higher degree of interactions between players and actions in a larger-scale simulation. As long as players responded dynamically, executed valid actions, and respected game constraints, it indicated that the system functioned as intended.

Results

Unlike the test cases conducted earlier, directly observing a CSV file for a 10x10 grid is hard to assess. Instead, a video recording was made using the Cell-DEVS Viewer, which visually demonstrates the state changes between cells, allowing us to see in real-time player decisions and ball interactions.

A link to the recording can be found here (inside the videos folder; 10x10_full): https://github.com/yelfaram/Cadmium-Cell-DEVS-Football-Player-Interaction

Discussion

The recorded simulation provides a clear way to analyze the interactions in the 10x10 grid. Observing the video, we can see that players dynamically adjust their behaviors, execute specific behavior actions (short pass, long pass, dribble, move, hold), and respect game constraints (e.g. only one ball can be present on the pitch at any given moment).
Key observations include:

- Players successfully perform actions based on their surroundings.
- Ball possession transitions correctly between players for passes and between cells for dribbles.
- Player positions transition correctly between cells for dribbles and moves.

This confirms that the behavior seen in the smaller 3x3 grid translates correctly when scaling to a 10x10 grid with more players.

Note: Refer to the README in the GitHub repository for details on what to look for in the simulation

Conclusion

The football player interaction model represented as a Cell-DEVS model and implemented using Cadmium, successfully demonstrated short passing, long passing, and dribbling interactions within a defined cell space. By scaling from a 3x3 grid to a 10x10 grid, the growing complexity of the dynamic interplay between players and the ball was evident. The model effectively captured player decision-making through various actions, showcasing how a modular approach using delayed propagation can result in state updates without modifying neighboring cells. This approach ensured that cell state modifications were isolated from one another.

It is important to note that the current model has limitations in its accuracy and realism. Currently, the decision-making process relies heavily on thresholds, and action costs drive the dynamic interaction. To enhance complexity, external factors that control mental and fatigue fluctuations need to be introduced, along with considerations for the current state of the game. Additionally, limiting the scope to just one team is not very reflective of real-life scenarios. Introducing another team or stationary obstacles could increase the realism of player actions such as dribbling or passing, as they would try to avoid them. On that note, the current von Neumann neighborhood limits dribbling options, restricting player movements to only horizontal and vertical directions. Adopting a hexagonal neighborhood can allow for a higher realism degree of player interactions, increasing the flexibility of player decisions.

Further improvements can be made by assigning weighted probabilities for actions alongside the thresholds to allow for a more dynamic decision-making process. By incorporating player-specific role logic, such as playmakers passing the ball more frequently and wingers dribbling the ball more often, we can enhance the model by making it less constrained and more reflective of real-life behavior. By improving the model's applicability in these areas, more complex simulations can be introduced.

Despite these limitations, the model provides a strong foundation for simulating player interactions within a cell-space grid and demonstrates how Cell-DEVS can be effective in showcasing spatial simulation. All in all, with further refinement in the aforementioned areas, the model has the potential to offer more insights into systems in dynamic environments such as sports.

References

[1] A. Makarenko, D. Krushinski, A. Musienko, and B. Goldengorin, “Towards Cellular Automata Football Models with Mentality Accounting,” *Lecture Notes in Computer Science*, pp. 149–152, 2010, doi: https://doi.org/10.1007/978-3-642-15979-4_16.