Farm Simulation Project Report

Ali Ayhan Günder
Student ID: 2021400219

CmpE 260 - Principles of Programming Languages
Bogazici University

5 June 2025

1 Introduction and Ideation Process

When I first read the project description, I felt both excited and worried. The farm simulation looked complex with many moving parts - agents that move, eat food, and reproduce. My first step was to understand the provided code files carefully. I spent time reading `cmpefarm.pro` and `farm.pro` to understand how the game mechanics work.

My main idea was to break down the big problem into smaller parts. I decided to start with simple predicates like calculating distance between agents, then move to more complex ones like pathfinding. This approach helped me build confidence as I progressed through the project.

2 Implementation Steps

2.1 Step 1: Basic Predicates (1-3)

I started with the easiest predicates. The `agents_distance/3` was straightforward - just Manhattan distance formula. For `number_of_agents/2`, I used `dict_pairs/3` to get all agents and count them. The `value_of_farm/2` required me to sum values of both agents and objects, which I did using `findall/3` and `sum_list/2`.

2.2 Step 2: Finding Functions (4-6)

These predicates required more thinking. For `find_food_coordinates/3`, I needed to check which foods the agent can eat using `can_eat/2`. The nearest agent and nearest food predicates required calculating distances to all candidates and sorting them. I learned that `sort/2` in Prolog automatically sorts by the first element, which was very useful.

2.3 Step 3: Pathfinding (7-8)

The `move_to_coordinate/6` was the most challenging part. I decided to use depth-first search (DFS) because it seemed simpler than other algorithms. My approach tracks the actions taken and reverses them at the end to get the correct path. I prevent infinite loops by not repeating the same action, though I realize this isn't perfect.

2.4 Step 4: Complex Behavior (9)

The `consume_all/6` predicate combines everything. I used a loop that:

- Finds nearest food
- Moves to it
- Eats it
- Repeats until no more food or depth limit reached

3 Challenges Encountered

Challenge 1: Understanding Dictionaries
I had never used Prolog dictionaries before. At first, I tried to access fields like `Agent.x` directly in patterns, which didn't work. I learned I need to use `get_dict/3` to extract values. This was frustrating but taught me to read documentation more carefully.

Challenge 2: Pathfinding Bugs
My first version of DFS had many bugs. Sometimes it would find very long paths or get stuck. I fixed this by:

- Adding the base case to check if already at destination
- Making sure to check if moves are valid before recursing
- Using `\+ member(Action, Path)` to avoid repeating moves

Challenge 3: Test Cases
Some test cases failed initially because I made wrong assumptions. For example, I calculated the wrong total value for the farm because I forgot wolves have no value. Debugging these issues helped me understand the problem better.

Challenge 4: Reproduction During Consumption
I worried about what happens when an agent reproduces during `consume_all`. The new child agent might block paths. I decided to keep my solution simple and accept this limitation, using the cut (!) to prevent backtracking.

4 Lessons Learned

Technical Lessons:

1. **Prolog is different** - I learned to think declaratively instead of imperatively. Instead of telling Prolog *how* to find something, I describe *what* I want to find.
2. **Backtracking is powerful but tricky** - Sometimes Prolog finds solutions I didn't expect. Using cuts and `once/1` helped control this.
3. **Built-in predicates save time** - Functions like `findall/3`, `sort/2`, and `member/2` are very powerful and saved me from writing complex code.

Problem-Solving Lessons:

1. **Start simple** - Beginning with easy predicates built my confidence
2. **Test frequently** - I tested each predicate immediately after writing it
3. **Read the specifications carefully** - Many of my initial bugs came from misunderstanding requirements

Personal Growth:
This project pushed me out of my comfort zone. I usually program in languages like Python or Java, so Prolog's logic programming paradigm was new to me. I learned to appreciate how elegant Prolog can be for certain problems, especially those involving search and constraints.

5 Conclusion

This farm simulation project was challenging but rewarding. I successfully implemented all required predicates and learned a lot about logic programming. While my solution isn't perfect (especially the pathfinding could be more efficient), it works correctly for all test cases.

If I had more time, I would improve the pathfinding to track visited positions and optimize the `consume_all` strategy to handle reproduction better. Overall, I'm satisfied with my work and grateful for the opportunity to learn Prolog in such an interesting way.