Parallel Functional Programming Final Project Proposal

Jonathan Chen (jyc2183), Kevin Wang (kjw2169)

Fall 2025

1 Introduction

Pac-Man is a classic arcade game that consists of a single player looking to navigate a board and consume all of the "pellets" present, while avoiding up to four ghosts that chase the player around the grid. Once a board is cleared of all "pellets", then the current level is considered "cleared" and the ghosts progressively become faster and faster. There are also power ups that allow the player to eat the ghosts, and special fruit on the board that are worth additional points for the player's score.

2 Problem

Objective: Full clear the board in the shortest number of moves/get highest possible score.

Room for increasing scope - moving up levels and resetting the board?

Lives: Pac-Man has one life - game ends when he runs into a ghost.

Ghosts (TBD): - All have same pathfinding - Some ghosts have different pathfinding - Deterministic (shortest path to pacman) - Nondeterministic (some aspect of randomness)

For simplification, no powerups and no fruits with bonus points (unless we need to increase scope).

Our input data will consist of a text file containing grid (maze) information, Pac-Man's starting location, the ghosts' starting locations, and their movement probabilities (if we choose to do a stochastic model).

Ex.

```
1 = wall
0 = empty cell
G = ghost
P = pacman
```

1	1	1	1	1	1	1	1	1	1
1	0	0	G	0	0	0	0	0	1
1	1	1	0	0	0	0	0	0	1
1	0	0	0	0	0	0	G	0	1
1	0	0	1	1	1	0	0	0	1
1	0	G	0	0	0	1	1	G	1
1	0	0	P	0	0	0	0	0	1
1	1	1	1	1	1	1	1	1	1

3 Approach

3.1 Sequential

If we move forward with a deterministic Pac-Man game, we will use the Minimax algorithm with alpha-beta pruning. We will alternate between maximizing Pac-man's score over his possible moves and minimizing the score between the combined possible moves of the four ghosts. We will use alpha-beta pruning to prune irrelevant branches.

If we instead decide to incorporate aspects of non-determinism, we will use Monte Carlo Tree Search (MCTS). We will optimize Pac-Man's four moves based on the expected value of each option. Essentially, this means we consider all possible randomized states when making each move, and calculate our potential score based on those states.

3.2 Parallelization

Deterministic Pac-Man game: parallelize the evaluation of Pac-Man's four moves: up, down, left, right. While this may affect our ability to properly prune branches, the overall time should still be dramatically reduced. Each move will be evaluated in its own spark, using Haskell's Strategies and rpar/rseq to evaluate each branch in our decision tree.

Stochastic Pac-Man game: The greatest computational demand is the simulation step where we calculate the probabilities of the ghost's movements and simulate the outcome of the game. We will parallelize this using Haskell's Strategies library, using rpar/rseq to evaluate simulations across available cores. Unfortunately, the backpropagation step needs to remain sequential, which will limit our overall speedup.

References

[1] Wikipedia - Monte Carlo tree search [online] Available at `https://en.wikipedia.org/wiki/Monte_Carlo_tree_search` [Accessed 15 Nov. 2025]

[2] Wikipedia - Pac-Man [online] Available at `https://en.wikipedia.org/wiki/Pac-Man` [Accessed 15 Nov. 2025]