Parallel Functional Programming Proposal: MazeSolver

Samya Ahsan (ska2138), Nicole Lin (nsl2126), Alice Wang (aw3271)

Fall 2023

1 Background & Objective

Rat in a Maze is a game set up as a maze on a two-dimensional nxn grid, where certain cells are marked as True (representing paths) and others as False (representing walls). The goal is to find the shortest distance to navigate a 'rat' from the start point $(0, 0)$ to the end point $(n-1, n-1)$ in the maze. The output is a list containing the shortest path found by the maze solver, represented as a list of positions (x, y).

2 Approach

Our approach involves finding a path through a maze from a starting position to a goal position. The maze is represented as a 2D grid, and each processor concurrently explores different paths within the maze, allowing for substantial parallelism.

The *Control.Parallel.Strategies* module is utilized to express parallelism conveniently. The 'par' is utilized to evaluate multiple paths concurrently. This enables the solver to efficiently navigate through the maze and discover multiple paths simultaneously.

3 Key Components

1. The maze is represented as a $2D$ list of Booleans, where True represents an open space, and False represents a wall. Positions within the maze are denoted as pairs of integers (x, y).
2. Represent the rat's position as a pair of coordinates (x,y).
3. Create functions that move the rat up, down, left, or right. Ensure that these movements are valid (not going out of bounds and not moving into walls).

4. The main exploration function recursively explores paths using backtracking. Parallel strategies are employed to distribute the exploration of multiple paths across processors.

    a Spawn different threads for different possible paths using parallelization *('par')*. For instance, if the rat can move both right and up, you can evaluate these moves in parallel.

    b Paths are extended by exploring next positions, ensuring that positions are not revisited to prevent infinite loops.

5. Return when the endpoint has been reached. The returning thread will be the shortest path.

This parallelized maze solver should show improved performance, especially on large mazes, by efficiently utilizing multiple processing units.