Parallel Functional Programming Final Project Proposal

Project Team Members: Rose Huang (rh2805) and Biqing Qiu (bq2134)

Description: Our final project will be to create a crossword solver for a crossword board with no hints. Given a board with blanks and blacked out boxes, we will search for different combinations of letters in the blank spaces using a brute force method and then introduce parallelism. We will have a dictionary of valid words to check our result against once we have a potential solution.

Possible Strategies: There are several different ways we could choose to approach finding a solution:

- Searching all combinations of letters (O^n different solutions, where n is the number of blank spaces): using depth-first search, fill in the given board with a possible combination of letters and check if it solves the board row-wise and column-wise. Possible optimization: checking before the whole board is filled so we can prune branches with known invalid words.
- Fitting words of the right length to board: constraint satisfaction problem. Starting with a word of the longest wanted length, we add a word from the dictionary with the right length into the block and backtrack to see if a solution satisfying all constraints can be found.

Verifying Solution: Once we have a board with completely filled letters, we can check by row and by column to see if the solution that we have arrived at is valid. For each row or column, we would split the characters into words by the blacked out boxes and check if each word is in the dictionary. This can be done in polynomial time.

Determinism: We could ensure that there is only one solution for the crossword to make the result deterministic by choosing a board configuration where there is only one combination of letters that create valid words for all the rows and columns. We could also provide filled in letters along with blacked out spaces, or verify against a small dictionary with only a subset of all english words.

Parallelizing the Solver: In the tree search, at every node which branches off into possible letters, we parallelize this branch so the search is done concurrently with other branches. For instance, with a 2x2 board. We could start off filling in the top right corner with all different letters of the english alphabet, A-Z, in parallel. Once the first blank is filled in, the search for the next letter could be done concurrently as well.
In the constraint satisfaction, backtracking, which also involves choosing from a selection of options and then recursing, can be parallelized.