COMS 4995 Parallel Functional Programming
Project Proposal : A.I. Hangman

Anthony Pitts
{UNI: aep2195}

Fall 2021

1 Project Abstract

For my Parallel Functional Programming final project, I will implement a version of the game "Hangman." My implementation will use AI heuristics to predict the next best letter choice in each round of the game. This will involve a significant amount of dictionary/ word processing, AI predictions, and Monadic operations that will lend itself very well to parallelism.

2 Explanation of this Hangman Version

Unlike other versions of Hangman, the attempts of letter choices will not be limited to the 5 or so "body parts" of the hangman. Instead, the program will run until the correct word has been discovered by the AI. The reason behind this difference is to demonstrate that the AI is actually working towards the goal of finding the word, and it is not simply making 5 incorrect guesses to reduce the runtime. Instead, the better the algorithm and use of parallelism, the faster the program will solve the word puzzle.

3 Explanation of the AI Algorithm

Each round of the game, the program must make a letter selection that it believes could be a missing letter in the word being guessed. My algorithm will use the heuristic of the "most common" possible letter. In other words, it will consider all words that have the same length as the word being guessed, and those with letters in the same positions as the correct letters already guessed. Of these words, it will select the letter that appears most frequently amongst them (excluding letters already guessed). This will obviously be a very computationally heavily operation in a large dictionary, which will lend itself well to "chunking" up the word processing in a parallel manner. I will also consider using a pruning

heuristic to stop processing words/letters if one of the letters appears to be an obvious first choice relative to the other letters' frequencies.

4 How I Plan to Parallelize the Program

In order to parallelize this algorithm, I will initially use parallelism on the IO operations when reading in the dictionary of words. I believe the parBuffer will work well here. Also, parallelism will be heavily involved in the selection of a letter each round. When finding the "most common" letter, it will have to process a very large amount of words and letters. For this reason, I plan to use "chunking" strategies on the frequency counting of all the words, utilizing all of the cores in my computer. I'm sure there will be other opportunities to parallelize in an application of this size, but those are just some stand-out areas that I'm sure will take advantage of parallelism.