Report

Challenge 1

For challenge 1, I used recursion to move the ray through the black box checking at each step if any atoms are in the vicinity. Guards were used to give precedence to absorption instead of deflection.

Challenge 2

For challenge 2, I opted for a brute force approach, finding every single combination of atoms, sorting them by increasing length, and using recursion to test one by one until the first answer is found. Because Haskell is a lazy language, when one answer is found, the recursion stops.

Challenge 3

For challenge 3, I decided firstly to convert the lambda expression to its alpha normal form, and then convert it to a string. Using recursion, I get deeper into the expression, changing any bound variables to the smallest existing integer available. If any bound variables are changed, so is every occurrence of the variable within the expression.

Challenge 4

For challenge 4, similarly to the parsers taught in the lectures, I implemented a series of five parsers which strictly follows the rules of the given language. Before the string runs through the parsers, a simple filter is used to remove all whitespaces.

Challenge 5

For challenge 5, I thought of the arithmetic expression as a tree structure and used recursion to go deeper into each node of the arithmetic expression given, converting it into a lambda expression at each step, until I reach the leaves of the tree.

Challenge 6

For challenge 6, I opted for a similar approach with challenge 5, thinking of the arithmetic and lambda expressions as tree structures, checking which nodes are the deepest, in order to implement an innermost leftmost reduction, which was then done by using recursion, to get to the deepest leaf.

Bibliography

[1] Anon, "Stack Overflow," 14 9 2015. [Online]. Available: https://stackoverflow.com/questions/32575630/powerset-of-a-set-with-list-comprehension-in-haskell. [Accessed 15 12 2022].

[2] Wikipedia the free encyclopedia, "Church encoding," [Online]. Available: https://en.wikipedia.org/wiki/Church_encoding. [Accessed 10 12 2022].

[3] J. Rathke, "PROGRAMMING III (COMP2209) - Lecture 37," 2022. [Online]. Available: https://secure.ecs.soton.ac.uk/notes/comp2209/22-23/37-LambdaCalculus.pdf. [Accessed 20 12 2022].

[4] J. Rathke, "PROGRAMMING III (COMP2209) - Lecture 38," 2022. [Online]. Available: https://secure.ecs.soton.ac.uk/notes/comp2209/22-23/38-ImplementingBeta.pdf. [Accessed 20 12 2022].

[5] J. Rathke, "PROGRAMMING III (COMP2209) - Lecture 45," 2022. [Online]. Available: https://secure.ecs.soton.ac.uk/notes/comp2209/22-23/45-Parsing.pdf. [Accessed 23 12 2022].