Lecture Summary: Functional Programming and Combinatorial Algorithms

Generated by LectureMate

February 22, 2025

1 Introduction

This lecture covered various topics in functional programming, particularly focusing on the implementation of combinatorial algorithms in Haskell. The session began with a brief overview of the programming project related to sequent calculus, followed by a discussion on combinatorial problems and their solutions.

2 Programming Project Overview

2.1 Sequent Calculus

The programming project involves implementing a function in Haskell that computes a list of simple sequences leading to a given sequence in sequent calculus. The project requires understanding the rules of sequent calculus, which were previously taught.

2.2 Project Requirements

- Implement a function that takes a sequence and returns a list of simple sequences.
- Handle implications and by implications as an optional extension.
- A challenge part involves optimizing proofs to minimize the number of steps.

2.3 Testing and Submission

Students are encouraged to submit their work multiple times for testing. The project is graded out of four marks, with specific tests designed to evaluate the implementation.

3 Combinatorial Algorithms

The lecture transitioned into combinatorial algorithms, focusing on generating combinations and permutations.

3.1 Choosing k Elements from a List

The problem of choosing k elements from a list of length n was discussed. The number of ways to choose k elements is given by the binomial coefficient, denoted as $\binom{n}{k}$.

3.2 Recursive Implementation

A recursive function was presented to compute combinations:

```
choose k [] = []
choose 0 xs = [ [] ]
choose n (x:xs) = choose (n-1) xs ++ map (x:) (choose (n-1) xs)
```

This function generates all combinations of length k from the list.

3.3 Partitions of a Number

The concept of partitions was introduced, where the goal is to find all the ways to express a number as a sum of positive integers. A recursive approach was suggested for generating these partitions.

4 Change-Making Problem

The change-making problem involves determining the different ways to make change for a given amount using a specified set of coins. The algorithm sorts the coins and recursively explores combinations that sum to the target amount.

5 The Eight Queens Problem

The final topic was the famous eight queens problem, where the objective is to place eight queens on a chessboard such that no two queens threaten each other.

5.1 Generate and Test Algorithm

A generate-and-test approach was used, where permutations of queen placements are generated, and each configuration is checked for validity:

```
validPlacement queens = all (not . canAttack) [(q1, q2) | (q1:qs) <- tails queens, q2
```

This function checks if any two queens can attack each other based on their positions.

5.2 Conclusion

The lecture concluded with a discussion on the efficiency of the algorithms presented and their practical applications in functional programming. The next lecture will cover monads, a key concept in Haskell.