Lecture Summary: Logic and Haskell

Generated by LectureMate

February 22, 2025

1 Introduction

Good afternoon everyone. Before we start, Don has asked me to remind you that if you want to change your tutorial group, do not email him, me, Kendall, or Vito. The only way to change your tutorial group is by filling in the tutorial change group request form.

2 Course Overview

Last week, we discussed bits, binary encodings, and propositional logic. We briefly touched on predicate logic, which we will not cover in detail due to its complexity. Instead, we will focus on philosophical logic from a modern perspective.

2.1 Philosophical Logic

We will explore examples using a universe defined by shapes with various properties such as size, color, and shape. The universe is a set of objects we will analyze.

3 Categorical Propositions

We will work with categorical propositions, which have the form:

- Every A is B
- Some A is B
- No A is B

These statements can be evaluated as true or false based on the universe we define.

3.1 Truth Evaluation

To determine the truth of these statements, we can use observation and search for counterexamples. For example:

- "Every red triangle is small" can be verified by checking for red triangles that are not small.
- "Every small triangle is red" can be disproven by finding a small triangle that is not red.

4 Formalization of Statements

We will formalize our reasoning process using logical notation. For instance, the statement "Every red triangle is small" can be expressed as:

$$\forall x \, (x \text{ is red} \implies x \text{ is small})$$

4.1 Predicate Logic

We will use a restricted form of predicate logic to express our categorical propositions. The structure of these statements allows us to apply logical reasoning systematically.

5 Haskell and Logic

We will integrate Haskell programming into our study of logic. Haskell is well-suited for implementing logical concepts and reasoning.

5.1 Defining Properties in Haskell

In Haskell, we can define properties of objects using predicates. For example, we can define a predicate for color:

```
isGreen :: Thing -> Bool
isGreen x = ...
```

5.2 List Comprehensions

Haskell's list comprehensions allow us to express logical statements succinctly. For example, to check if every small triangle is red, we can write:

```
all isRed [x | x <- things, isSmall x, isTriangle x]
```

This expression generates a list of truth values for the property of being red among small triangles.

6 Conclusion

In the upcoming weeks, we will continue to explore the intersection of logic and programming in Haskell. We will formalize our understanding of categorical propositions and implement them in Haskell to enhance our logical reasoning skills.