A Computer-Checked Library of Category Theory

Universal Properties of Category Theory in Functional Programming

1 - Background

"Category theory is the mathematics of mathematics. Whatever mathematics does for the world, category theory does for mathematics" - E. Cheng [1]

- **Category theory** provides a general framework for describing mathematical concepts by focusing on relationships rather than the internal details of the objects (using **universal properties**)
- **Computer proof assistants** are software tools for formalizing mathematical concepts

2 - Project Goals and Motivation

Goal	Motivation
Implement a pedagogical library of category theory in Lean	Existing category theory libraries are too advanced for beginners
Answer the research question: *Which parallels can be drawn between the universal properties of category theory and functional programming?*	Lack of papers highlighting and exemplifying the connection between universal properties of category theory and functional programming

3 - Methodology

PHASE 1

Learn about **category theory** and **Lean**

↓

Implement **core features** together with the group

↓

PHASE 2

Extend the library by defining the **universal properties**

↓

Answer the **research question**

4 - Implementation

Category consists of

- **Objects**
- **Morphisms**
- **Identity morphisms** (Id)
- **Composition operator** ($\circ$)

Category must satisfy

- **Left unit law:** $f \circ Id = f$
- **Right unit law:** $Id \circ f = f$
- **Associative law:** $(h \circ g) \circ f = h \circ (g \circ f)$

Category was implemented as a structure containing a field for each component and axiom

Implemented universal properties:

- **Initial objects** have a unique morphism to every object
- **Terminal objects** have a unique morphism from every object
- **Binary products** represent the combinations of two objects along with projections
- **Binary coproducts** represent the combinations of two objects along with injections

Each universal property includes a function checking this property, structure representing an object with this property and an example in the Set category

The main goal was to make the library as beginner-friendly as possible so we focused on

- writing **intuitive code**
- including **concrete examples**
- adding a lot of **documentation**

5 - Findings

Found parallels between the implemented **universal properties** and **functional programming**:

Universal Property	FP Concept	Similarities and Insights
Initial objects	Empty types	• Impossible to define a terminating function to an empty type • Exactly one function from the empty type to any type (the empty function)
	Base cases in recursion	• Simplest (and often emptiest) starting points for constructing more complex elements • Each recursive data type has a unique construction/representation
Terminal objects	Unit types	• No point in defining functions from a unit type • Useful to return a unit type for functions that perform side-effects
Binary products	Product types	• Set of possible values is the cartesian product of the sets of the possible values of its field types • Field accessors and argument extraction in pattern matching are similar to projections
Binary coproducts	Sum types	• Set of possible values is the disjoint union of the sets of the possible values of its field types • Constructors and case handling in pattern matching are similar to injections

All parallels also included examples from **Haskell**:

`data Void` ← **Empty type**

`data () = ()` ← **Unit type**

Sum type **Product type**

`data Shape = Circle Float | Rectangle Float Float`

`data List a = Nil | Cons a (List a)`

Base case in recursion

6 - Discussion

Limitations:

- Not as general/abstract
- Not as many features

Possible future improvements:

- Add **more universal properties** + examples
- Add **more examples** of categories
- **Prove theorems** about universal properties
- **Find parallels** between other universal properties and functional programming

References

[1] E. Cheng, *How to bake pi: An edible exploration of the mathematics of mathematics.* Basic Books, 2015.

CSE3000 Research Project

Markus Orav
M.Orav@student.tudelft.nl

Professor: Benedikt Ahrens
Supervisor: Lucas Escot

TUDelft