EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Mathematisch-Naturwissenschaftliche Fakultät

Programmiersprachen und Softwaretechnik

Prof. Klaus Ostermann

Leitung des Seminars
David Binder
Ingo Skupin

Kategorientheorie für Programmierer

Hausaufgabenblatt Abschlussprojekte – SS18

Tübingen, 29. Januar 2020

1. Wir haben im Seminar sowohl mit Funktoren als auch mit Monaden programmiert. „Zwischen“ diesen beiden Typklassen gibt es noch die Typklasse `Applicative` die in dem Artikel „Applicative programming with effects“ von Conor McBride und Ross Paterson eingeführt wurde.

   Das Projekt besteht in der Vorstellung des Programmierens mit Applicatives und dem Verhältnis zur Kategorientheorie.

2. Monaden lassen sich zu sogenannten „Arrows“ erweitern, welche von John Hughes in „Generalizing Monads to Arrows“ vorgestellt wurden. Für das Projekt gilt das obengesagte für Applicatives.

3. Zum Parsen werden häufig sogenannte „monadische Parserkombinatoren“ wie in den Bibliotheken „Parsec“, „Attoparsec“ oder „Megaparsec“ verwendet. Diese erlauben es auf einfache Weise komplexe Parser aus modularen kleineren Parsern zusammenzubauen.

4. Das duale Konzept zu Monaden sind Comonaden, welche in Kapitel 23 des Buches vorgestellt werden.

5. Ein vielversprechender Ansatz zur Strukturierung funktionaler Programme sind die sogenannten „freien Monaden“. Freie Monaden erlauben es eine Menge von Operationen (z.B. Daten aus einer Datenbank auslesen oder Text auf die Konsole schreiben) zu definieren. Programme können dann diese beiden Operationen verwenden, und Interpreter erlauben es die Operationen verschieden zu interpretieren: Beispielsweise mit einer gestubbten Datenbank in einem Testkontext oder als Anfrage an eine echte Datenbank in einem Produktionskontext.

6. Eine weitere interessante Monade ist die sogenannte *Probability Monade* Dist T, die als Typ von Wahrscheinlichkeitsverteilungen über T interpretiert wird. Mit der Probability Monade können zum Beispiel auf sehr einfache Weise Simulationen von Spielen und Gewinnwahrscheinlichkeiten implementiert werden.

7. Wir haben gesehen dass aus jeder Adjunktion eine Monade hervorgeht. Es gilt auch die umgekehrte Richtung in folgender Form: Für jede Monade $T : \mathcal{C} \to \mathcal{C}$ können zwei verschiedene Adjunktionen angegeben werden: Die Adjunktionen mit der Kleisli-Kategorie $\mathcal{C}_T$ und der Eilenberg-Moore Kategorie $\mathcal{C}^T$.

   Das Projekt besteht in der Vorstellung dieser beiden Konstruktionen.

8. Noson Yanofsky's „A Universal Approach to Self-Referential Paradoxes, Incompleteness and Fixed Points“ zeigt, dass zunächst verschieden erscheinende Phänomene wie das Russell'sche Paradox der Menge aller Mengen die sich nicht selbst enthalten, das Halteproblem, die Überabzählbarkeit der reellen Zahlen uvm. als Instanz des gleichen Phänomens in Kategorien mit Produkten und Exponentials (= cartesian closed categories) erklären lassen.

9. Continuation-Passing-Style (CPS) ist ein wichtiger Bestandteil der Theorie und Implementierung von funktionalen Programmiersprachen. Das Projekt besteht in der Vorstellung von Continuation-Passing-Style (CPS) und der CPS-Monade in Haskell.

10. Beweis und Anwendungen des Yoneda-Lemmas.
11. Anstelle von kommutativen Diagrammen gibt es noch eine andere graphische Notation für Kategorien: String Diagramme. Das Projekt besteht in der Vorstellung von String Diagrammen für gewöhnliche sowie für monoidale Kategoren.
12. Ein klassisches Beispiel für eine Kategorie ist die Kategorie der Mengen und Funktionen zwischen Mengen. Viele Eigenschaften dieser Kategorie sind uns schon bekannt: Sie hat Produkte und Koprodukte, Exponentials sowie natural number objects. In „Rethinking Set Theory“ stellt Tom Leinster vor wie man die Kategorie der Mengen nur durch solche rein kategoriellen Eigenschaften bestimmen kann.
13. Vorstellung von Lawvere Theorien als Alternative zu Monaden.
14. Kan Extensions, Ends, Coends . . .
15. . . .