EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Mathematisch-Naturwissenschaftliche Fakultät

Programmiersprachen

Prof. Klaus Ostermann

Leitung des Seminars
David Binder
Ingo Skupin

Kategorientheorie für Programmierer

Hausaufgabenblatt 6 – WS19

Tübingen, 4. Dezember 2019

Aufgabe 1: Lektüre

Für die kommende Woche lesen Sie bitte Abschnitte 1 bis 4 (inklusive) des Artikels „Reason Isomorphically!“ von Ralf Hinze und Daniel W. H. James.[1]

Aufgabe 2: Yoneda-Lemma in Aktion

In dieser Aufgabe geht es um einen ganz besonderen Isomorphismus zwischen Typen, den wir in gemischter Mathematik-Haskell Notation so schreiben:

$$(\texttt{forall}\ x.(a \to x) \to fx) \simeq fa$$

Hierbei sollen f ein Funktor und a ein konkreter Typ sein. Auf der linken Seite des Isomorphismus steht der Typ der in x polymorphen Funktionen von $a \to x$ nach fx, und auf der rechten Seite des Isomorphismus steht ein konkreter Typ fa.

Wir wollen diesen Isomorphismus in Haskell nachimplementieren. Damit das funktioniert benötigen wir die Spracherweiterung `RankNTypes`. Dann sehen die Typen der beiden Teile des Isomorphismus folgendermaßen aus:

```
{-# LANGUAGE RankNTypes #-}
f1 :: Functor f => (forall x.((a -> x) -> f x)) -> f a
f1 = undefined
f2 :: Functor f => f a -> (forall x.((a -> x) -> f x))
f2 = undefined
```

- Implementieren Sie `f1` und `f2`. Hinweis: Obwohl die Haskell-Typen kompliziert aussehen, brauchen Sie trotzdem keine neue Syntax, um `f1` und `f2` zu implementieren. Sobald Sie eine Lösungsidee haben, probieren Sie es einfach aus. Der Compiler erkennt die richtige Lösung automatisch an.
- Setzen Sie für f den Maybe-Funktor ein und für a den Unit Typ $()$. Zeigen Sie, dass daraus folgt, dass genau 2 polymorphe Funktionen vom Typ `x -> Maybe x` existieren.
- Setzen Sie für f den Listen-Funktor ein und für a den Typ `Integer`. Wie interpretieren Sie den Isomorphismus, der dabei herauskommt?

[1] Für manche ist eventuell auch dieser nlab-Artikel hilfreich, indem die „High-Level“ Idee des Yoneda-Lemmas erklärt wird (man kann hier im Text *space* einfach immer durch *object* ersetzen). Interessant sind hier vor allem die Abschnitte 2–4: `https://ncatlab.org/nlab/show/motivation+for+sheaves%2C+cohomology+and+higher+stacks#maps_between_generalized_spaces`

- Finden Sie eine interessante Kombination von einem Funktor und Typ den Sie kennen, setzen Sie die beiden in obige Gleichung ein, und berichten Sie uns Ihre Beobachtung :)

Aufgabe 3: Isomorphismen

Seien $\mathcal{C}$, $\mathcal{D}$ und $\mathcal{I}$ Kategorien und $F\colon \mathcal{C} \to \mathcal{D}$ ein volltreuer (= *fully faithful*) Funktor.

1. Zeigen Sie, dass der Postkompositionsfunktor $F\circ\colon \mathcal{C}^{\mathcal{I}} \to \mathcal{D}^{\mathcal{I}}$ zwischen den beiden Funktorkategorien $\mathcal{C}^{\mathcal{I}}$ und $\mathcal{D}^{\mathcal{I}}$, definiert durch $F\circ(G) = F\circ G$ auf Objekten und $F\circ(\eta) = F\eta$ auf Morphismen, ebenfalls volltreu ist.
2. Geben Sie einen Funktor an, der voll ist, aber keine Isomorphismen reflektiert.
3. Geben Sie einen Funktor an, der treu ist, aber keine Isomorphismen reflektiert.