EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

**Mathematisch-Naturwissenschaftliche Fakultät**
Wilhelm-Schickard-Institut

```
unrollVec : ∀ {n} → (v : V.Vec (P.∃₂ Text) n)
          → V.Vec Char (V.sum (V.map P.proj₁ v))
unrollVec V.[] = V.[]
unrollVec (x V.:: x₁) = P.proj₁ (P.proj₂ x) V.++ unrollVec x₁

concat : ∀ {n} → (x : V.Vec (P.∃₂ Text) n) → Text _ _
concat = vec→text ∘ unrollVec

concatMap : ∀ {n v} → (f : Char → P.∃₂ Text) → Text n v → Text _ _
concatMap {v = v} f _ = concat (V.map f v)
```

# Kategorientheorie für Programmierer

## Einführung und Organisation des Seminars

**16. April 2018, David Binder, Ingo Skupin**

# Inhaltsverzeichnis

# Kategorientheorie

Kategorientheorie

- ist ein Bereich der Mathematik der (wie die Mengentheorie) in der gesamten Mathematik Anwendung findet.
- wurde 1945 von Samuel Eilenberg und Saunders MacLane begründet.
- untersucht Strukturen die in allen mathematischen Teilgebieten auftauchen. („Refactoring“ der Mathematik.)

$\Rightarrow$ Hoher Grad von Abstraktion.

# Kategorientheorie

Was haben die folgenden Dinge gemeinsam?

- Kartesisches Produkt von 2 Mengen: $\{1, 2, 3\} \times \{a, b, c\}$
- Produkt von 2 Monoiden: $M_1 \times M_2$
- Konjunktion in der Logik: $\phi \wedge \psi$

Alles sind Instanzen eines (kategoriellen) Produkts.

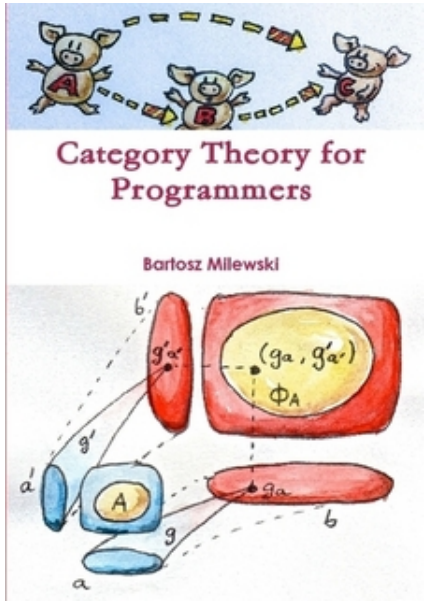
# Funktionale Programmierung und Kategorientheorie

Wieso ist Kategorientheorie anwendbar in der funktionalen Programmierung?

- Kategorientheorie ist die Algebra von Funktionen, und dem Hintereinanderausführen von Funktionen.
- Die zentrale Idee der funktionalen Programmierung ist das Komponieren von simplen zu komplexen Funktionen.
- Objekte werden nicht durch ihre innere Struktur beschrieben, sondern über ihr „funktionales Interface“.

# Seminarlektüre I

Wir lesen das Buch „Category Theory for Programmers“ welches kostenlos online als pdf verfügbar ist. Alternativ kann man für ca 20 Euro eine Print-on-Demand Version auf lulu.com bestellen. (Lange Lieferzeiten!)



# Seminarlektüre II

Das Seminar findet wöchentlich statt. Jede Woche

- werden 1-2 Kapitel aus dem Buch gelesen. Gegen Ende lesen wir evtl. auch andere Artikel.
- übernimmt ein Student die Rolle des Diskussionsleiters.
- bereitet sich jeder auf das Seminar vor indem der Text zu Hause gelesen wird.

# Haskell

Im Seminar verwenden wir die Programmiersprache Haskell für die Beispiele.

- Vorkentnisse in Haskell sind **nicht** notwendig und können im Laufe des Seminars nachgeholt werden.
- Jeder sollte eine lauffähige Version von Haskell (ghc/ghci) installiert haben.
- Wenn Fragen zu Haskell auftauchen die Ihr nicht im Seminar stellen wollt könnt ihr uns gerne auch direkt kontaktieren.

# Übungsaufgaben

Es wird jede Woche ein kurzes Übungsblatt geben.

- Das Bearbeiten der Übungsblätter ist nicht verpflichtend und wird nicht benotet.
- Das Bearbeiten der Übungsblätter wird dringend empfohlen. Nur indem man selbst Beispiele durchgeht bekommt man ein Gefühl für den Inhalt der Definitionen.
- Besprechung der Übungsblätter jeweils zu Beginn des Seminars.

# Final Project

Gegen Ende des Seminars wird es Final Projects geben in denen Ihr die gelernten Grundlagen nutzt um euch in eine Anwendung der Kategorientheorie einzulesen.Themen können aus den folgenden Bereichen gewählt werden:

- Programmiertechniken
- Programmiersprachentheorie
- Theoretische Informatik
- Mathematik und Logik
- . . .

Das Final Project besteht entweder aus Code oder aus einer kurzen schriftlichen Ausarbeitung und einer Präsentation.

# Benotung

Es werden folgende Leisungspunkte im Bereich praktische Informatik vergeben:

- 3 LP für Master in der neuen PO
- 3 LP für Master in der alten PO / 4 LP für die alte PO falls als SQ angerechnet wird.
- Bachelor können sich nach Regelung der PO auch Masterveranstaltungen anrechnen lassen.

Die regelmäßige aktive Teilnahme am Seminar ist Voraussetzung für das Bestehen des Seminars. Die Endnote setzt sich wie folgt zusammen:

- 100% Final Project.

# Danke.

Kontakt: David Binder, Ingo Skupin

**Wilhelm-Schickard-Institut**
Programmiersprachen und Softwaretechnik
Sand 14, 72076 Tübingen
Büro: B211
Telefon: +49 7071 29-70516
binderd@informatik.uni-tuebingen.de
skupin@informatik.uni-tuebingen.de