



# Praktische Informatik 3: Funktionale Programmierung

## Vorlesung 8 (06.12.2022): Übungen

Christoph Lüth



Deutsches  
Forschungszentrum  
für Künstliche  
Intelligenz GmbH



Universität  
Bremen

Wintersemester 2022/23

# Was zum Nachdenken

## Übung 8.1: Import

Warum schreibt man

```
import Prelude hiding (repeat)
```

und was bewirkt das? (Hinweis: `Prelude` ist das Modul der vordefinierten Funktionen.)

# Was zum Nachdenken

## Übung 8.1: Import

Warum schreibt man

```
import Prelude hiding (repeat)
```

und was bewirkt das? (Hinweis: `Prelude` ist das Modul der vordefinierten Funktionen.)

**Lösung:** Die Import-Anweisung importiert alle vordefinierten Funktionen **bis auf** `repeat`.  
Dadurch können wir `repeat` selber (anders) definieren.

## Übung 8.2: Die Map als Assoziativliste

```
data Map  $\alpha$   $\beta$  = Map [( $\alpha$ ,  $\beta$ )]  
insert :: Ord  $\alpha$   $\Rightarrow$   $\alpha \rightarrow \beta \rightarrow$  Map  $\alpha$   $\beta \rightarrow$  Map  $\alpha$   $\beta$   
insert a b m = (a,b):m
```

Was ist der Nachteil dieser einfachen Implementation?

## Übung 8.2: Die Map als Assoziativliste

```
data Map  $\alpha$   $\beta$  = Map [( $\alpha$ ,  $\beta$ )]  
insert :: Ord  $\alpha$   $\Rightarrow$   $\alpha \rightarrow \beta \rightarrow$  Map  $\alpha$   $\beta \rightarrow$  Map  $\alpha$   $\beta$   
insert a b m = (a,b):m
```

Was ist der Nachteil dieser einfachen Implementation?

**Lösung:** Erzeugt ein Speicherleck — überschriebene Elemente bleiben in der Liste. Besser: beim Einfügen alte Elemente entfernen

```
insert :: Ord  $\alpha$   $\Rightarrow$   $\alpha \rightarrow \beta \rightarrow$  Map  $\alpha$   $\beta \rightarrow$  Map  $\alpha$   $\beta$   
insert a b xs = (a, b): filter ((a  $\neq$  ).fst) xs
```

Nicht sehr effizient. Besser: Map als **sortierte** Liste.

# Was zum Selbermachen

## Übung 8.3: Use the Source, Luke!

Ladet euch von der Webseite der Veranstaltung die Quellen für die 8. Vorlesung herunter, und öffnet die Datei `MapTree.hs`.

Vergleicht die Haskell-Implementation mit den Beschreibung der Folien.

Welche der Funktionen `lookup`, `insert`, `delete` könnte man als `fold` realisieren?

# Was zum Selbermachen

## Übung 8.3: Use the Source, Luke!

Ladet euch von der Webseite der Veranstaltung die Quellen für die 8. Vorlesung herunter, und öffnet die Datei `MapTree.hs`.

Vergleicht die Haskell-Implementation mit den Beschreibung der Folien.

Welche der Funktionen `lookup`, `insert`, `delete` könnte man als `fold` realisieren?

Lösung: `lookup` läßt sich falten:

```
lookup' k = fold ( $\lambda$ ak ax l r  $\rightarrow$  if k == ak then Just ax  
                  else maybe r Just l) Nothing
```

Ist aber nicht so effizient (linear statt logarithmisch), weil es immer erst links, dann rechts sucht.