Prof. Dr. Manfred Schmidt-Schauß
Künstliche Intelligenz/Softwaretechnologie

Fachbereich Informatik (20)
Johann Wolfgang Goethe-Universität Frankfurt am Main

# Funktionale Programmierung I

Sommersemester 2000

# Übungsblatt Nr. 8

Abgabe: Dienstag 30. Mai 2000 vor der Vorlesung

## Aufgabe 1 (15 Punkte)

Definieren Sie durch $\lambda$-freie Superkombinatoren in KFP+

1. den Fixpunktkombinator `Y`, so daß `Y f` $\sim_c$ `f (Y f)` gilt.
2. mittels Lambda-Lifting folgende KFP-Ausdrücke:
   (a) $\lambda z.\lambda y.((\lambda z.z)\ y)$
   (b) $\lambda f.(\lambda x.(\lambda h.h\ (h\ x))f)$

## Aufgabe 2 (10 Punkte)

Schreiben Sie eine Funktion `mapmap` in der Form

```
mapmap f xs = map (\x -> ...) xs
```

die eine Funktion `f` und eine Liste von Listen als Eingabe hat und die Funktion `f` auf jedes Element jeder Unterliste anwendet: z.B. soll `mapmap quadrat [[1], [2,3]]` zu `[[1], [4,9]]` reduzieren.

Wenden Sie Lambda-Lifting auf diese Definition an und geben Sie das Ergebnis in möglichst einfacher Form an.

## Aufgabe 3 (10 Punkte)

Übersetzen Sie die fibonacci-Funktion nach KFP. Sie sei in KFP+ gegeben durch die Definition

$$\texttt{fib}\ n\ =\ \texttt{case}_{\texttt{Int}}\ n\ (0\ \texttt{->}\ (S\ 0))\ (S\ 0\ \texttt{->}\ (S\ 0))\ (S\ (S\ m)\ \texttt{->}\ (\texttt{fib}\ (S\ m)) + (\texttt{fib}\ m))$$

## Aufgabe 4 (15 Punkte)

Programmieren Sie eine Funktion `combine :: (a -> b -> c) -> (d -> a) -> (d -> b) -> d -> c` in KFP+ (oder Haskell), die das Ergebnis der Anwendung zweier Funktionen `f1 :: (d -> a)` und `f2 :: (d -> b)` auf jeweils ein Argument `x :: d` mittels `op :: (a -> b -> c)` verknüpft.

`combine and even odd n` z.B. ergäbe `False` für jede Zahl n.