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

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

Einführung in das funktionale Programmieren:

Sommersemester 1997

Übungsblatt Nr. 4

Aufgabe 1

In Haskell ist es möglich "list comprehensions" zu verwenden. Geben Sie an wie diese nach **KFP** übersetzt werden und zeigen Sie, daß die Übersetzung korrekt ist.

Aufgabe 2

Eine Möglichkeit **KFP** Ausdrücke in Haskell darzustellen verwendet Datenkonstruktoren für die syntaktischen Elemente von **KFP**. Z.B.

```
data KFPexp
        = VarConst String
        | App String
        | Case [(String, Int)] KFPexp [KFPexp]
```

Um mit einem `Case` Konstruktor die verschiedenen $\texttt{case}_A$ aus **KFP** darzustellen, hat dieser als erstes Argument die Liste der Konstruktoren von A mit deren Stelligkeit. Die folgenden beiden Argumente entsprechen dann den Argumenten des $\texttt{case}_A$.

- Schreiben Sie eine Funktion, `pprKFP`, in Haskell, die solche Ausdrücke "schön" ausgibt.
- Schreiben Sie eine Funktion, `nextRedex`, die in einem solchen Ausdruck den nächsten auszuwertenden Ausdruck bei normaler Reihenfolge ermittelt und entsprechend kenntlich macht. (Hinweis: `nextRedex` soll als erstes Argument eine Liste erhalten, in der die im Programm bekannten Funktionsnamen zusammen mit ihrer maximalen Stelligkeit angegeben sind.)

(wird fortgesetzt)

Aufgabe 3

Wir erweitern den λ-Kalkül der Vorlesung um arithmetische Operationen und positive ganze Zahlen. In **KFP** kennen wir entsprechende Funktionen z.b. `PLUS`, `TIMES`, ... Übersetzen sie den λ-Ausdruck

$$\lambda f.\lambda x.\lambda h.(\lambda z.((/\ z)\ h)\ (\lambda y.((-\ (f\ y))\ (f\ x))\ (\lambda m.((-\ m)\ h)\ x)))$$

mittels Einführung expliziter Funktionsnamen und Lambda-Lifting in ein **KFP** Programm.

Aufgabe 4

In Haskell kann `concat` definiert werden als

$$\texttt{concat} = \texttt{foldr}\ (\texttt{++})\ \texttt{[]}$$

Warum ist diese Definition gegenüber der Definition

$$\texttt{concat} = \texttt{foldl}\ (\texttt{++})\ \texttt{[]}$$

vorzuziehen? Begründen Sie Ihre Antwort.