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. 10

Aufgabe 1

Im λ-Kalkül besitzen wir nicht die Fähigkeit wie in **KFP**, Funktionen an Namen binden zu können. Angenommen wir hätten eine „Standardumgebung", in der Funktionen wie `+, -, *, /, = , !=, ...`, das $\texttt{case}_A$ wie wir es von **KFP** her kennen, natürliche Zahlen, `True` und `False` bekannt seien. Wir könnten dann Fakultät *nicht* einfach rekursiv wie folgt definieren:

$$\text{fak} = \lambda n.(\texttt{case true}\ (= n\ 1)\ 1\ (*\ n\ (\text{fak}\ (-\ n\ 1)))),$$

wobei die Zuordnung fak $= \ldots$ hier als reines Text-Makro zu verstehen ist. Würden wir das Makro fak expandieren, erhielten wir einen unendlichen String.

a) Zeigen Sie, daß $Y = \lambda f.(\lambda x.(f\ (x\ x))\ \lambda x.(f\ (x\ x))$ ein Fixpunktoperator ist, d.h. für jeden beliebigen λ-Ausdruck e gilt:

$$(Y\ e) = e\ (Y\ e)$$

b) Benutzen Sie den Fixpunktoperator Y aus Aufgabenteil a), um die rekursive Funktion fak in eine gültige λ-Version zu überführen und berechnen Sie mit Ihrer Version das Ergebnis von 2!.

Aufgabe 2

a) Versuchen Sie mit den Typregeln aus dem Script das folgende **KFP**-Programm zu typisieren (dabei ist die Funktion `head` wie gewöhnlich auf Listen typisiert):

```
twice f x = f (f x)
dbhead = twice head
```

b) Schreiben Sie eine Funktion `dbhead'`, die dieselbe Semantik wie `dbhead` hat, aber typisiert werden kann.

c) Wie ist der Typ von `dbhead'` ?

Aufgabe 3

a) Versuchen Sie zwei Konstruktoren `Nil'` und `Cons'` für Listen so zu typisiern, daß sie Listen konstruieren können, die in einer Liste Elemente verschiedener Typen enthalten können. (Dazu müssen die Konventionen der Vorlesung möglicherweise erweitert werden.)

b) Typisieren Sie für Listen aus Teil a) die Funktionen `head` und `length`.

c) Geben Sie eine Funktion `f` an, die Sie typgerecht auf einen Ausdruck `(head list)` für eine Liste aus Teil a) anwenden können.