

Einführung in die funktionale Programmierung

Wintersemester 2007/2008

Aufgabenblatt Nr. 3

Abgabe: Dienstag 6. November 2007 **vor!** der Vorlesung

Aufgabe 1 (10 Punkte)

Definieren Sie in Haskell eine Funktion `laengste :: [[a]] -> [a]`, die eine Liste von Listen erwartet und nur die längsten Unterlisten zurückliefert.

Z.B. sollte die Eingabe

`laengste ["Das", "längste", "Wort", "in", "diesem", "Text", "ist", "sieben", "Zeichen", "lang"]`
als Ausgabe die Liste `["längste", "Zeichen"]` ergeben.

Aufgabe 2 (15 Punkte)

Implementieren Sie in Haskell eine Funktion `rechtsbuendig :: Int -> String -> String`, die eine Zeilenbreite (als Anzahl von Zeichen) sowie einen Text als String erwartet und diesen entsprechend der Zeilenbreite *rechtsbündig* setzt. Falls Zeilen länger als die Zeilenbreite sind, soll ein Fehler generiert werden (hierfür gibt es die vordefinierte Funktion `error :: String -> a`).

Ein Beispiel für Ein- und Ausgabe ist im folgenden abgebildet:

```
Habe nun, ach! Philosophie,  
Juristerei und Medizin,  
Und leider auch Theologie  
Durchaus studiert, mit heißem Bemühn.  
Da steh ich nun, ich armer Tor!  
Und bin so klug als wie zuvor;  
Heiße Magister, heiße Doktor gar  
Und ziehe schon an die zehen Jahr  
Herauf, herab und quer und krumm  
Meine Schüler an der Nase herum-  
Und sehe, daß wir nichts wissen können!  
Das will mir schier das Herz verbrennen.
```

Eingabe txt

```
Habe nun, ach! Philosophie,  
Juristerei und Medizin,  
Und leider auch Theologie  
Durchaus studiert, mit heißem Bemühn.  
Da steh ich nun, ich armer Tor!  
Und bin so klug als wie zuvor;  
Heiße Magister, heiße Doktor gar  
Und ziehe schon an die zehen Jahr  
Herauf, herab und quer und krumm  
Meine Schüler an der Nase herum-  
Und sehe, daß wir nichts wissen können!  
Das will mir schier das Herz verbrennen.
```

Ausgabe des Aufrufs `rechtsbuendig 55 txt`

Aufgabe 3 (25 Punkte)

Gegeben sei die folgende Datentyp-Definition¹ für den Datentyp `EXPR` in Haskell, die zur Darstellung von KFPT-Ausdrücken verwendet werden soll, wobei wir annehmen, es gäbe nur die Typen `Bool`, `List` und `Paar` mit den entsprechenden Konstruktoren:

```
data EXPR = Variable Var
          | Lambda Var EXPR
          | App EXPR EXPR
          | CTrue
          | CFalse
          | CCons EXPR EXPR
          | CNil
          | CPaar EXPR EXPR
          | CaseBool EXPR [Alt]
          | CaseList EXPR [Alt]
          | CasePaar EXPR [Alt]
    deriving (Show, Eq)

type Alt = (Pat, EXPR)
type Pat = EXPR
type Var = String
```

- Implementieren Sie in Haskell eine Funktion `istKFPTAusdruck :: EXPR -> Bool` die für einen gegebenen Ausdruck vom Typ `EXPR` prüft, ob dieser ein KFPT-Ausdruck ist. (6 Punkte)
- Implementieren Sie zwei Funktionen `betaReduktion :: EXPR -> EXPR` und `caseReduktion :: EXPR -> EXPR` die für einen KFPT-Ausdruck eine *unmittelbare* β - bzw. `case`-Reduktion durchführen, falls dies möglich ist und ansonsten das Argument unverändert als Ergebnis liefern. Im Quellcode auf der WWW-Seite ist bereits eine Funktion `substitute` definiert, die hilfreich sein könnte. Des weiteren gehen wir davon aus, dass die Ausdrücke bereits korrekt umbenannt sind, so dass Sie das Einfangen von Variablen nicht berücksichtigen müssen. (7 Punkte)
- Implementieren Sie ein Prädikat `istWHNF :: EXPR -> Bool`, das für einen KFPT-Ausdruck prüft, ob dieser eine WHNF ist. (2 Punkte)
- Implementieren Sie die Einschnitt-Normalordnungsreduktion für KFPT-Ausdrücke, indem Sie zunächst nach dem passenden Reduktionskontext suchen, und anschließend die Funktionen aus Aufgabenteil b) anwenden. (6 Punkte)
- Verwenden Sie Ihre Lösungen aus Aufgabenteil d) und e) um die Normalordnungsreduktion zu implementieren. D.h. definieren Sie eine Funktion `noReduktion :: EXPR -> EXPR`, die einen KFPT-Ausdruck solange in Normalordnung reduziert, bis eine WHNF erreicht ist und diese dann als Ergebnis zurückliefert. (4 Punkte)

¹Der Quellcode ist auch auf der WWW-Seite zur Veranstaltungen zu finden