

Einführung in die funktionale Programmierung

Wintersemester 2007/2008

Aufgabenblatt Nr. 1

Abgabe: Dienstag 23. Oktober 2007 **vor!** der Vorlesung

Bitte senden Sie den zu Ihrer Lösung zugehörigen dokumentierten Quellcode auch per Email an `sabel@ki.informatik.uni-frankfurt.de`

Aufgabe 1 (12 Punkte)

Betrachten Sie folgende Konstrukte:

1. $((\backslash x \rightarrow (\text{Cons True})) y)$
 2. $(\text{if True then (Paar True True) else (Paar False False)})$
 3. $(\text{case}_{\text{List}} ((\backslash z \rightarrow z) \text{Nil}) \text{ of } \{\text{Nil} \rightarrow (\text{Paar } a b); (\text{Cons } a b) \rightarrow ((\backslash x \rightarrow (\text{Paar } x x)) b)\})$
 4. $(\text{Cons } ((\backslash x \rightarrow (x x)) (\backslash y \rightarrow (y y))) ((\backslash u \rightarrow (\backslash z \rightarrow u)) \text{Nil}) \text{True})$
 5. $(\text{case}_{\text{Paar}} x \text{ of } \{(\text{Paar } y y) \rightarrow \text{True}; (\text{Paar } y z) \rightarrow \text{False}\})$
 6. $((\backslash x \rightarrow (\text{Cons True } x \text{ True})) \text{True})\text{T}$
 7. $((\backslash z \rightarrow (z x)) (\text{case}_{\text{Paar}} a \text{ of } \{(\text{Paar } a b) \rightarrow ((\backslash x \rightarrow (x y)) a)\}))$
 8. $((\backslash x \rightarrow x) (\text{case}_{\text{List}} (\text{Cons True Nil}) \text{ of } \{(\text{Cons } a \text{ Nil}) \rightarrow a; \text{Nil} \rightarrow \text{False}\}))$
- a) Welche der Konstrukte sind KFPT-Ausdrücke? Begründen Sie, warum die übrigen Konstrukte keine KFPT-Ausdrücke sind. (8 Punkte)
- b) Berechnen Sie für die KFPT-Ausdrücke, jeweils die *gebundenen* und *freien* Variablen. Welche der KFPT-Ausdrücke sind *geschlossen*? (4 Punkte)

Aufgabe 2 (15 Punkte)

Gegeben sei der folgende KFPT-Ausdruck t :

$$t = \left(\left(\left(\left(\text{case}_{\text{Bool}} (((\backslash x \rightarrow ((\backslash z \rightarrow z) (\backslash y \rightarrow y))) \text{False}) \text{True}) \right) \right) \right) \left(\left(\begin{array}{l} \text{of } \{\text{False} \rightarrow (\backslash a \rightarrow a); \\ \text{True} \rightarrow ((\backslash u \rightarrow (\backslash v \rightarrow v)) \text{True})\} \end{array} \right) (\backslash b \rightarrow b) \right) \text{False} \right)$$

- a) Zeichnen Sie den zum Ausdruck t zugehörigen Termgraphen, wobei:
- Abstraktionen werden durch einen mit „ λ “ markierten Knoten dargestellt, der zwei Kinder besitzt: Das linke Kind ist die durch die Abstraktion gebundene Variable, das zweite Kind entspricht dem Rumpf der Abstraktion.
 - Applikationen werden durch einen mit „ $@$ “ markierten Knoten dargestellt, der zwei Kinder für den Ausdruck an Funktionsposition und dem Argument besitzt.
 - Konstruktoranwendungen haben als Knotenbeschriftung den Konstruktornamen, und n Kinder, wenn der Konstruktor Stelligkeit n besitzt.
 - **case**-Ausdrücke haben $n+1$ Kinder, wenn n die Anzahl der Alternativen ist: Das erste Kind ist das erste Argument des **case**-Ausdrucks, die anderen Kinder entsprechen den Alternativen.
 - eine **case**-Alternative „Pattern“ \rightarrow „Ausdruck“ wird durch einen mit \rightarrow markierten Knoten dargestellt, der zwei Kinder besitzt: einen für das Pattern und einen für die rechte Seite der Alternativen. (10 Punkte)
- b) Geben Sie **alle** Reduktionskontexte R mit $\exists s : R[s] = t$ an. (5 Punkte)

Aufgabe 3 (23 Punkte)

Üben Sie die Benutzung des Interpreters GHCi oder HUGS, nachdem Sie diesen installiert bzw. auf den Rechnern der RBI gestartet haben.

- a) Lassen Sie den Ausdruck aus Aufgabe 2 in einem der Interpreter auswerten. Beachten Sie, dass Sie den Ausdruck direkt im Interpreter eingeben können, wenn Sie das Konstrukt `case_Boolean` durch `case` ersetzen. (3 Punkte)
- b) Implementieren Sie eine Funktion `ggT` die den *größten gemeinsamen Teiler* zweier natürlichen Zahlen berechnet, indem Sie diese in einer Datei mit der Endung `.hs` definieren und im Interpreter mit `:load Dateiname` laden.
Rufen Sie die Funktion `ggT` für einige Werte auf.
Welche Information erhalten Sie, wenn Sie im Interpreter `:t ggT` eingeben? (9 Punkte)
- c) Implementieren Sie eine Funktion `comb` in Haskell, die eine binäre Operation `b`, zwei einstellige Operationen `u1`, `u2` sowie zwei Ausdrücke `w1`, `w2` erwartet und zunächst `u1` auf `w1` und `u2` auf `w2` anwendet und schließlich diese Ergebnisse mittels der binären Operation `b` zum Gesamtergebnis verbindet.
Z.B. ergibt `comb (+) (\x -> x*3) (\x -> x `div` 2) 10 20` als den Wert 40. (7 Punkte)
- d) Implementieren Sie zwei Operationen `nand` und `nor`, die das logische Nicht-Und bzw. Nicht-Oder berechnen, wobei Sie *ausschließlich* die Funktion `comb` sowie die eingebauten Operationen `not` (logisches Nicht), `&&` (logisches Und) und `||` (logisches Oder) verwenden sollten. (4 Punkte)