Einführung in die funktionale Programmierung

Wintersemester 2016/2017

Aufgabenblatt Nr. 3

Abgabe: Montag, 7. November 2016 vor der Vorlesung

Bitte schicken Sie die Haskell-Quelltexte auch per Email an
hanczak@ki.informatik.uni-frankfurt.de

Aufgabe 1 (15 Punkte)

Die folgenden KFPT-Ausdrücke seien gegeben:

a) $(\lambda y.((\lambda x.x)\ \texttt{True}))$ 1,2,4,5,9

7,10 b) $((\lambda y.\texttt{case}_{\text{List}}\ y\ \texttt{of}\ \{(\texttt{Cons}\ a\ b) \to b;\ \texttt{Nil} \to \texttt{Nil}\})\ \texttt{True})$

4,6,9 c) $((\lambda z.\texttt{case}_{\text{Bool}}\ z\ \texttt{of}\ \{\texttt{True} \to \texttt{True};\ \texttt{False} \to (\texttt{True}\ \texttt{Nil})\})\ \texttt{True})$

10 d) $(\texttt{case}_{\text{Paar}}\ (\texttt{Paar}\ \texttt{True}\ (\lambda z.(z\ z)))\ \texttt{of}\ \{(\texttt{Paar}\ a\ b) \to (b\ b)\})$

7,8,10 e) $(\texttt{case}_{\text{Bool}}\ (\lambda x.\texttt{True})\ \texttt{of}\ \{\texttt{True} \to \texttt{False};\ \texttt{False} \to \texttt{True}\})$

1,3,4,6,9 f) $(\texttt{Cons}\ (\texttt{True}\ \texttt{True})\ (\texttt{Cons}\ \texttt{True}\ (\texttt{Cons}\ \texttt{True}\ \texttt{Nil})))$

Σ = 19/50

Geben Sie für jeden der obigen Ausdrücke an, welche der folgenden Aussagen auf ihn zutreffen.

1. Der Ausdruck *ist eine WHNF.*
2. Der Ausdruck *ist eine FWHNF.*
3. Der Ausdruck *ist eine CWHNF.*
4. Der Ausdruck *hat eine WHNF.*
5. Der Ausdruck *hat eine FWHNF.*
6. Der Ausdruck *hat eine CWHNF.*
7. Der Ausdruck *ist dynamisch ungetypt.*
8. Der Ausdruck *ist direkt dynamisch ungetypt.*
9. Der Ausdruck *terminiert.*
10. Der Ausdruck *divergiert.*

Für die Lösung der Aufgabe reicht eine Tabelle von folgender Form aus:

	1	2	3	4	5	6	7	8	9	10
a)	X	X		X	X				X	
b)	X	X		X	X		X			X
c)	X	X		X	X	X			X	
d)	X		X	X	X	X				X
e)	X		X	X	X	X	X	X		X
f)	X		X	X		X	X	X	X	X

✓

9/15

Dabei sollte angekreuzt sein, welche Ausdrücke die entsprechende Eigenschaft erfüllen.

zu 2b) beta Reduce a = a 5/5

c) case Reduce (Case Bool (Bool True) b c) = b

case Reduce (Case Bool (Bool False) b c) = c

Case Reduce a = a

caseList ?

2/5

Aufgabe 2 (35 Punkte)

Der folgenden Datentyp `Expr` stellt KFPT-Ausdrücke dar, die die Typen List und Bool verwenden. Er ist polymorph über dem Typ für die Variablen definiert:

```
data Expr a =
   Var a                                      -- x       = Var "x"
 | App (Expr a) (Expr a)                      -- (e1 e2) = App e1 e2
 | Lam a (Expr a)                             -- \x.e    = Lam "x" e
 | ListCons (Expr a) (Expr a)                 -- a:as    = ListCons a as
 | ListNil                                    -- []      = ListNil
 | BoolTrue                                   -- True    = BoolTrue
 | BoolFalse                                  -- False   = BoolFalse
 | CaseList (Expr a) (Expr a) (a,a,Expr a)-- case_List e of {[] -> e1; (x:xs) -> e2)
                                              -- = CaseList e (e1) ("x",xs",e2)
 | CaseBool (Expr a) (Expr a) (Expr a)     -- case_Bool e of {True -> e1; False -> e2)
                                              -- = CaseBool e e1 e2
  deriving(Eq,Show)
```

Auf der Webseite zur Vorlesung finden Sie den Haskell-Quellcode für den Datentyp, einige Beispiele und eine Funktion `substitute`, welche die Substitution durchführt (`substitute e1 e2 x` berechnet gerade `e1[e2/x]`).

Implementieren Sie in Haskell

a) drei Funktionen `isFWHNF`, `isCWHNF` und `isWHNF`, die einen KFPT-Ausdruck erwarten und prüfen, ob dieser eine FWHNF, eine CWHNF bzw. eine WHNF ist. (Jede der Funktionen hat den Typ `Expr a -> Bool`). (5 Punkte)

b) die Funktion `betaReduce :: Expr String -> Expr String`, die einen KFPT-Ausdruck erwartet und diesen – falls möglich – *unmittelbar* β-reduziert. Ist dies nicht möglich, so soll die Eingabe unverändert zurück gegeben werden. (5 Punkte)

c) die Funktion `caseReduce :: Expr String -> Expr String`, die einen KFPT-Ausdruck erwartet und diesen – falls möglich – *unmittelbar* case-reduziert. Ist dies nicht möglich, so soll die Eingabe unverändert zurück gegeben werden. (5 Punkte)

d) die Funktion `isUntyped :: Expr String -> Bool`, die einen KFPT-Ausdruck erhält und prüft, ob dieser *direkt dynamisch ungetypt* ist. (7 Punkte)

e) die Funktion `oneStepNormalOrder :: Expr String -> Expr String`, die einen KFPT-Ausdruck erhält und einen Normalordnungsreduktionsschritt für den Ausdruck ausführt. Ist kein solcher Schritt möglich, so soll die Eingabe unverändert zurück gegeben werden. (8 Punkte)

f) die Funktion `calcResult :: KFPTAusdruck -> Result`, die einen KFPT-Ausdruck erhält und diesen solange in Normalordnung auswertet, bis feststeht, ob der Ausdruck eine WHNF hat (in diesem Fall soll `TerminatesWith e` zurückgeliefert werden, wobei `e` die erhaltene WHNF ist) oder divergiert (in diesem Fall soll `Diverges` zurückgeliefert werden). Dabei sei `Result` der folgende Datentyp:

```
data Result = TerminatesWith (Expr String)| Diverges
 deriving(Eq,Show)
```

Die Funktion `calcResult` darf für getypte divergierende Ausdrücke nicht terminieren. (5 Punkte)

a) isFWHNF :: Expr a → Bool
isFWHNF (Lam _ _) = True ✓
isFWHNF _ = False

isCWHNF :: Expr a → Bool
isCWHNF (App _ _) = True
isCWHNF = False CWHNF ist was anderes

2 isWHNF :: Expr a → Bool
isWHNF a = (isFWHNF a) ✓ 3/5
|| (isCWHNF a)

b) betaReduce :: Expr String → Expr String
betaReduce (App (Lam a b) c) = BIT