WS 2017/2018

Funktionale Programmierung

12. Übungsblatt

Prof. Dr. Margarita Esponda

(Abgabetermin: Mi., den 31.01.2018, um 10:10, Hörsaal)

Ziel: Auseinandersetzung mit Lambda-Kalkül.

1. Aufgabe (12 Punkte)

Ergänzen Sie das **LCI** Modul aus der Vorlesung, das die Hauptfunktionen des Lambda-Kalkül Interpreters enthält, um folgende Hilfsfunktionen:

a) Deklarieren Sie den **Expr** algebraischen Datentyp als Instanz der **Show** Klasse, indem Sie auch die **show** Funktion selber definieren. Programmieren Sie dafür eine **expr2String** Funktion, die bei Eingabe beliebiger Werte des **Expr**-Datentyps diesen in eine lesbare Zeichenkette umwandelt, die entsprechende Lambda-Ausdrücke darstellt.
Das **λ**-Symbol wird mit Hilfe des **"/"** Zeichens dargestellt.

Anwendungsbeispiele:

expr2String (Var "x") ⇒ "x"

expr2String (Lambda "z" (Var "z")) ⇒ "/z.z"

expr2String (App (Lambda "x" (Lambda "y" (App (Var "x") (Var "y")))) (Lambda "z" (Var "z")))
⇒ "(/x./y.xy)(/z.z)"

b) Definieren Sie eine Funktion **string2Num**, die eine positive Zahl als Zeichenkette bekommt und diese in die entsprechende Zahl umwandelt.

Anwendungsbeispiel:

string2Num "234" ⇒ 234

c) Definieren Sie eine **nat2lambda** Funktion, die bei Eingabe einer natürlichen Zahl den entsprechenden Lambda-Ausdruck generiert.

Anwendungsbeispiel:

nat2lambda 7 ⇒ "(/s./z.s(s(s(s(s(s(z)))))))"

2. Aufgabe (9 Punkte)

a) Ergänzen Sie die **expand** Funktion aus der Vorlesung mit allen selbstdefinierten Funktionen des 10. und 11. Übungsblatts und programmieren Sie unter Verwendung der **expand** und **nat2lambda** Funktionen eine **expandExpr** Funktion, die bei Eingabe eines beliebigen Funktionsnamens (als Zeichenkette) einen entsprechend ausgeweiteten Lambda-Ausdruck zurückgibt.

Anwendungsbeispiel:

expandExpr "5" ⇒ "(/s./z.s(s(s(s(z)))))"

expandExpr "FIB" ⇒ "(/rn.{=}n{0}{0}({=}n{1}{1}({+}(r({P}n))(r({P}({P}n))))))"

3. Aufgabe (9 Punkte)

Schreiben Sie Testfunktionen für alle bis jetzt selbstdefinierten Lambda-Funktionen mit Hilfe des Lambda-Interpreters.

Wichtige Hinweise:

2. Schreiben Sie in allen Funktionen die entsprechende Signatur.
3. Verwenden Sie geeignete Namen für Ihre Variablen und Funktionsnamen, die den semantischen Inhalt der Variablen oder die Semantik der Funktionen wiedergeben.
4. Verwenden Sie vorgegebene Funktionsnamen, falls diese angegeben werden.
5. Kommentieren Sie Ihre Programme.
6. Verwenden Sie geeignete lokale Funktionen und Hilfsfunktionen in Ihren Funktionsdefinitionen.