Theorie und Implementation objektorientierter Programmiersprachen
Theoretische Informatik, LMU München
Andreas Abel

Übungsblatt 8

Abgabe bis: Montag, **1. Juli 2002, 10.15 Uhr**

Aufgabe 1: [Aliasing]

Eine Referenz s wird als *Alias* einer Referenz r bezeichnet, falls s die gleiche Speicherzelle addressiert wie r. Definieren Sie eine Funktion

$$\texttt{alias} \ : \ \texttt{Ref}\,\texttt{Bool} \to \texttt{Ref}\,\texttt{Bool} \to \texttt{Bool},$$

die entscheidet, ob die übergebenen Referenzen Aliase zueinander sind. Dabei darf der Inhalt der Speicherzellen überschrieben werden.
Beweisen Sie durch Angabe der Reduktionssequenzen die Korrektheit ihrer Funktion, d.h. zeigen Sie für Adressen $l \neq l'$

$$\begin{array}{lcl} \texttt{alias}\ l\ l \mid \mu & \longrightarrow^* & \texttt{true} \mid \mu' \\ \texttt{alias}\ l\ l' \mid \mu & \longrightarrow^* & \texttt{false} \mid \mu' \end{array}$$

Dabei ist μ ein beliebiger Speicherzustand, in dem die Adressen l und l' definiert sind.
(4 Punkte)

Aufgabe 2: [Simulation von Rekursion durch Referenzen]

Definieren Sie eine rekursive Funktion $\texttt{sum} : \texttt{Int} \to \texttt{Int}$

$$\texttt{sum}\, n = \sum_{i=1}^{n} i$$

mit Hilfe von `fix`. Definieren Sie nun eine gleichwertige Funktion $\texttt{sum}'$ *ohne* `fix` zu verwenden, sondern unter Zuhilfenahme einer *Referenz auf eine Funktion*.
Generalisieren Sie Ihre Methode und skizzieren Sie ein allgemeines Schema, wie man rekursive Funktionen $\texttt{fix}\, f : T.\ t$ mit Hilfe von Referenzen ohne `fix` darstellen kann.
(4 Punkte)

Aufgabe 3: [Veränderbare Datensätze (OOP I)]

Ein Punkt in der Ebene sei repräsentiert durch den Datensatz-Typ

$$\texttt{PointR} = \{x : \texttt{Ref Int},\ y : \texttt{Ref Int}\}.$$

Definieren Sie ein Funktion `create` : `Int` → `Int` → `PointR`, die einen neuen Punkt erzeugt und eine Funktion `move` : `PointR` → `Int` → `Int` → `Unit`, die den Punkt relativ in der Ebene verschiebt. Zeigen Sie durch Angabe der wesentlichen Schritte der Reduktionssequenz, dass

```
let p = create i 1
in (move p j -1; p.x)
```

zu `i+j` auswertet.
(4 Punkte)

Aufgabe 4: [Abstrakte Datentypen (OOP II)]

Ein Punkt in der Ebene unterstütze folgende Methoden, die in einem *Interface* zusammengefasst werden:

$$\begin{aligned}\texttt{PointI} = \{\ & \texttt{getX} : \texttt{Unit} \to \texttt{Int},\\ & \texttt{getY} : \texttt{Unit} \to \texttt{Int},\\ & \texttt{move} : \texttt{Int} \to \texttt{Int} \to \texttt{Unit}\ \ \}\end{aligned}$$

(Dies ist ein "ganz gewöhnlicher" Datensatz-Typ!) Definieren Sie einen Konstruktor `newPoint` : `Int` → `Int` → `PointI` für Punkt-Objekte. Zeigen Sie durch Angabe der wesentlichen Schritte der Reduktionssequenz, dass

```
let p = newPoint i 1
in (p.move j -1; p.getX ())
```

zu `i+j` auswertet.
(4 Punkte)

Viel Erfolg!