Karlsruher Institut für Technologie

Institut für Programmstrukturen und Datenorganisation

Prof. Dr.-Ing. Gregor Snelting	Prof. Dr. Ralf Reussner

gregor.snelting@kit.edu	reussner@kit.edu

Programmierparadigmen – WS 2022/23

https://pp.ipd.kit.edu/lehre/WS202223/paradigmen/uebung

Blatt 2: Bindung, Kombinatoren, Terme

Abgabe: 10.11.2022, 12:00
Besprechung: 14.11. – 15.11.2022

Reichen Sie Ihre Abgabe bis zum 10.11.2022 um 12:00 in unserer Praktomat-Instanz unter https://praktomat.cs.kit.edu/pp_2022_WS ein.

Geben Sie Ihre Lösung in Freitextform ab. Bäume können Sie z.B. eingescannt, abfotografiert oder als ASCII-Art abgeben.

1 Bindung und Gültigkeitsbereiche

Geben Sie für jede Verwendungsstelle jedes Bezeichners an, auf welche Definitions- bzw. Bindungsstelle er verweist. Hierzu bieten sich Pfeile an, wie in Zeile 1 gezeigt.

Hinweis: Probieren Sie an einem eigenen Beispiel aus, ob **let** oder **where** stärker bindet!

```
1 fun x = x
2 f y = \z -> x + 7 * z - y
3 x = 1
4 g x = x + (let y = x * 2; x = 5 * 5 in (let x = f x 2 in x + y))
5 h = let z = 2 in g x + (\z -> -z) z where z = 3
```

binding.hs

2 Listenkombinatoren

Geben Sie Ihre Lösungen als Modul `Polynom`[1] ab.

Polynome $a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}x^{n-1} + a_nx^n$ können durch Listen ihrer Koeffizienten dargestellt werden – zweckmäßigerweise mit a_0 am Listenanfang.

```
type Polynom = [Double]
```

Um mehrdeutige Repräsentationen zu vermeiden sei die kanonische Darstellung eines Polynoms die *kürzestmögliche*, also so dass z.B. das Nullpolynom als leere Liste dargestellt wird.

1. Definieren Sie eine Funktion `cmult` zur Multiplikation eines Polynoms mit einer Konstanten. Verwenden Sie `map`.

[1] Also in einer Datei `Polynom.hs` mit erster Zeile `module Polynom where`

$$\texttt{cmult :: Polynom -> Double -> Polynom}$$

2. Definieren Sie die Auswertung

$$\texttt{eval :: Polynom -> Double -> Double}$$

   eines Polynoms an einer Stelle x mit dem Horner-Schema. Verwenden Sie `foldl` oder `foldr`. Zu obigem Polynom ist das Horner-Schema

$$a_0 + x \cdot (a_1 + x \cdot (a_2 + \ldots (a_{n-1} + x \cdot a_n) \ldots))$$

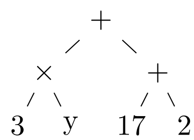
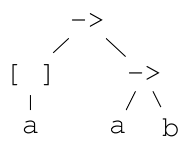
3. Definieren Sie eine Funktion `deriv :: Polynom -> Polynom` zum Ableiten eines Polynoms. Welcher Listenkombinator bietet sich an? Zu obigem Polynom ist die Ableitung

$$a_1 + 2a_2 \cdot x + 3a_3 \cdot x^2 + \ldots + n \cdot a_n \cdot x^{n-1}$$

B-Seite: Termsprachen

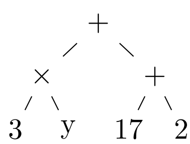
Termsprachen tauchen in dieser Vorlesung fast überall auf. Wer sie nicht als solche erkennt, muss immer wieder die gleichen Prinzipien aufs Neue erlernen, deswegen schärfen wir nun unser Gespür für sie.

Terme (auch *Ausdrücke*) sind *strukturierte Objekte*, die meist in linearer Stringdarstellung (potenziell mit Klammern) geschrieben werden. Ein paar Beispiele:

- $3 \times y + (17 + 2)$ ist ein Term, und hat die Struktur 
- `[a] -> (a -> b)` ist ein Term mit Struktur 
- $(3 \times y) + (17 + 2)$ und `[a] -> a -> b` sind andere Schreibweisen für die *selben* Terme von oben (Erinnerung: der Funktionstyp ist *rechtsassoziativ*!)
- $\forall x.\ (P(x) \land \forall y.\ Q(x))$ und $(\forall x.\ P(x)) \land (\forall y.\ Q(x))$ sind zwei *unterschiedliche* Terme. Geben Sie jeweils die Struktur an, der Allquantor ist hierbei ein binärer Operator.

Assoziativität und Priorität ("welche Klammern kann man weglassen?", "Punkt vor Strich", etc.) beeinflussen die Zuordnung zwischen Stringdarstellung und Term, aber die Struktur ist *inhärent*.

Manchmal müssen wir über die *Syntax* zweier Termsprache hinwegsehen, um Zusammenhänge zwischen scheinbar verschiedenen Termsprachen zu sehen. Denkbare Schreibweisen sind z.B.:

Infix	Präfix	Baum
$3 \times y + (17 + 2)$	$+(\times(3, y), +(17, 2))$	

1. Geben Sie folgende Terme in einer geeigneten Infix- (für den letzten Term), Präfix-, sowie Baum-Schreibweise an:
   - Den Haskell-Typen `a -> (a -> (b,b)) -> b`
   - Die Formel $V_{out} = \frac{R_2}{R_1+R_2} \times V_{in}$
   - Den balancierten Binärbaum mit 7 Knoten, in dessen Knoten jeweils die Höhe des Teilbaums steht.

Es ist insbesondere wichtig, die Termstruktur zu berücksichtigen, wenn wir Terme in andere Terme einsetzen.

2. Setzen Sie den Haskell-Term $X =$ `a -> b` in folgende Terme ein:
   - `c -> d ->` X
   - X `-> e -> f`
3. Stellen Sie die Terme nach Einsetzung mit möglichst wenig Klammern dar, falls nicht schon der Fall.
4. Übersetzen Sie die Ausgangsterme und Ihre Lösungen in die Baumdarstellung. War Ihre Einsetzung tatsächlich strukturerhaltend?