3. Übung Programmierung

Dominic Deckert

2. Mai 2017

Previously on ...

- Rekursion
- pattern matching

Nachzuholendes

Aufgabe 2:
b) Trenne Satz nach Leerzeichen auf
a) Füge Satz aus Wörtern zusammen

Aufgabe 3:
Ermittle Länge der längsten Liste

Bäume

Selbstdefinierung struktureller Datentypen möglich mittels data

```
Branch  (Branch  (Leaf 5)
                 (Branch  (Leaf 2)
                          (Leaf 2)))
        (Branch  (Leaf -3)
                 (Leaf 0))
```

Aufgabe 1

1.) Funktion, die Blätter zählt

2.) Blätterliste (von links nach rechts)

Polymorphe Datentypen

Polymorphe Datentypen

Datenstrukturen oder Funktionen können tw. mit variablem Datentyp beschrieben werden
→ Typenvariable (hier: a)

polymorpher Datentyp	Struktur mit variablem Datentyp
polymorphe Funktion	Funktion mit variablen Datentypen

Hinweis: In Haskell sind Funktionen auch Datentypen

Aufgabe 2

a) minimale Pfadlänge eines Baumes berechnen (Wurzel: 1)

b) Baumbeschriftung durch entsprechenden Pfad ersetzen

Wichtige Higher-Order-Funktionen

Wichtige Higher-Order-Funktionen

Higher-Order-Funktion: Funktion, die Funktion(en) als Eingabewert(e) hat

map	(a->b) -> [a] -> [b]	führt Funktion auf jedem Listenelement aus
filter	(a -> Bool) -> [a] -> [a]	Filtert Liste nach “True”-Elementen
foldr	(a -> b -> b) -> b -> [a] -> b	“Faltet” Liste von rechts auf

Aufgabe 3

Berechne Produkt der Quadrate der geraden Zahlen
Hinweis: even testet, ob eine Zahl gerade ist

Aufgabe 4

Implemetiere map für typpolymorphe Bäume: tmap