Wichtig: Snag-Heim "Grep" rausschreiben aus Video

funktionalität: ###

Zitate Heute: „Das ist "Bastard-Code"" -> Xander de Kiene

Funktionen höherer Ordnung (higher-order functions)

; Elemente der Liste xs, die Prädikat p? erfüllen:

```
(: filter ((%a -> boolean) (list-of %a) -> (list-of %a)))

(define filter
  (lambda (p? xs)
    (match xs
      (empty empty)
      (make-pair y ys (if (p? y)
                          (make-pair y (filter p? ys))
                          (filter p? ys))))))
```

- Wert des Parameters p? ist selber wieder eine Funktion
  => p? kann angewandt werden

Higher-order function (H.o.f.) ...

(1) akzeptieren Funktionen als Parameter und/oder

(2) liefern eine Funktion als Ergebnis

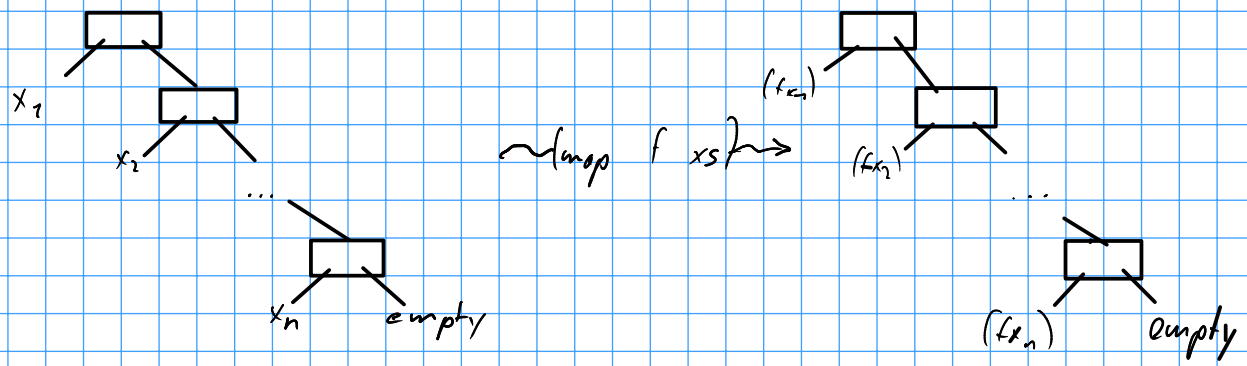
filter ist vom Typ (1)

H.o.f. vermeiden Duplizierungen von Code und führen zu

- kompakteren Programmen,
- verbesserter Lesbarkeit
- verbesserte Wartbarkeit

Bsp. (map f xs)

xs:



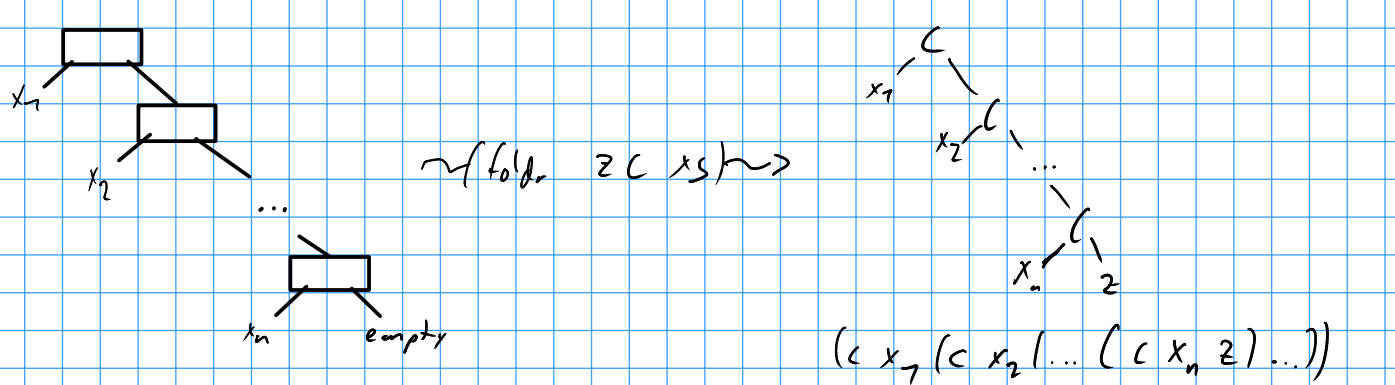
; wende f auf alle Elemente von xs an

```
(: map ((% a -> % b) (List-of % a) -> (List-of % b)))
(define map
 ((lambda (f xs)
  (match xs
   (empty empty)
   ((make-pair y ys) (make-pair (f y) (map f ys))))))))
```

Verwende einfache Lambda-Abstraktionen direkt als anonyme Funktionen, wenn eine globale Benennung (via define) nicht gerechtfertigt erscheint (z.B. nur lokale/einmalige Nutzung)

Allgemeinere Transformationen von Listen: Listenfaltung (list folding)

Idee: Ersetze die Listenkonstruktoren make-pair und empty systematisch:



- (folder z c xs) wirkt als Spare Transformer
  - empty -> z
  - make-pair -> c
- Eingabe : (list-of %a)
- Ausgabe : im allgemeinen <u>leere Liste</u> mehr etwa %b

; Falte Liste xs bzgl. z und c

(: folder (%b (%a %b -> %b) (list-of %a) -> %b))

(define folder
  (lambda (z c xs)
    (match xs
      (empty z)
      ((make-pair y ys) (c y (folder z c ys))))))