Tutorium - Informatik 1

—

Montags 16 st.
Jules Kreuer, VL bei Prof. Grust

Bewertung der Leistung

⚠ Eine Bewertung eurer Leistung in der *Informatik 1* wird allein auf Basis des Übungsbetriebes getroffen.
Wir werden **keine Klausur** zum Semesterende durchführen.

13 / 9 ÜB = 100%
14tes ÜB = Bonus

Allgemeine Infos:
https://forum-db.informatik.uni-tuebingen.de/t/informatik-1-keine-klausur-planungen-zum-weiteren-uebungsbetrieb/9016

Anmeldung zur Bewertung:
https://forum-db.informatik.uni-tuebingen.de/t/wichtig-fuer-alle-action-required-before-feb-1-2021-anmeldung-zur-pruefung-in-der-informatik-1/9048

Stand: 25.01 10:30

ÜB 8

HOF, foldr & foldr

Informatik 1

Forum: https://forum-db.informatik.uni-tuebingen.de/c/ws2021-info1

Übungsblatt 8 (13.01.2021)

Abgabe bis: Mittwoch, 20.01.2021, 14:00 Uhr

Relevante Videos: bis einschließlich `Informatik 1 - Chapter 08 - Video #035`.

https://tinyurl.com/Informatik1-WS2021

Die **Lehrevaluation für das WS 20/21 läuft noch bis Freitag, 18:00 Uhr**. Wenn ihr in eure Mailbox schaut, findet ihr dort eine Nachricht von unserer Studiendekanin die euch ermöglicht an der Lehrevaluation teilzunehmen. Wir wären euch sehr dankbar, wenn ihr euch ein paar Minuten Zeit nehmt und daran teil nehmt. Jede Rückmeldung, egal ob positiv oder negativ, hilft uns die Vorlesung *Informatik 1* noch besser zu gestalten. **Danke!**

Sprachebene „Die Macht der Abstraktion“

Aufgabe 1: [8 Punkte] **Wichtig:** In dieser Aufgabe müssen Fallunterscheidungen über Listen mit Hilfe von Pattern Matching (`match`) implementiert werden.

Programmiert eine *Funktion höherer Ordnung* `zipWith` mit der folgenden Signatur:

```
(: zipWith ((%a %b -> %c) (list-of %a) (list-of %b) -> (list-of %c)))
```

`(zipWith f xs ys)` kombiniert Elemente der Listen `xs` und `ys` an den gleichen Positionen mittels der Funktion `f`. Beachtet, dass überstehende Elemente bei der Verarbeitung verworfen werden.

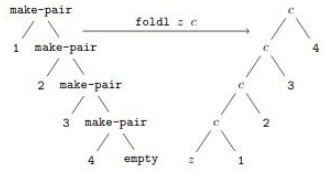
Beispiel:

```
(zipWith + (list 1 2 3) (list 10 20 30 40)) ⇝ (list 11 22 33)
```

Aufgabe 2: [10 Punkte] **Wichtig:** In dieser Aufgabe müssen Fallunterscheidungen über Listen mit Hilfe von Pattern Matching (`match`) implementiert werden.

In der Vorlesung wurde die Listenfaltung mittels `(foldr` z c xs`)` vorgestellt. Dabei wird die zwei-stellige Funktion c zuerst auf das letzte Element der Liste (und den Startwert z) angewendet, die Liste wird also gewissermaßen “von rechts” gefaltet.

Analog dazu lässt sich auch eine Listenfaltung “von links” `(foldl` z c xs`)` definieren. Dabei wird die zwei-stellige Funktion c zuerst auf das erste Element der Liste (und den Startwert z) angewendet. Ist die Liste leer, wird z zurückgegeben. Abbildung 1 illustriert die Faltung von links.



Schwierigkeiten ÜB 8

Aufgabe 1

- cons/touple zum matchen zweier Listen

Aufgabe 2

- Echt gut

Aufgabe 3

- zu wenig Tests

Aufgabe 4

- meist sehr gute gelöst. Kleinigkeiten wie Formatierung

Schwierigkeiten A1 ÜB 8

“Die Elementer zweier Listen mithilfer einer Funktion f verweben”
(: zipWith ((%a %b -> %c) (list-of %a) (list-of %b) -> (list-of %c)))

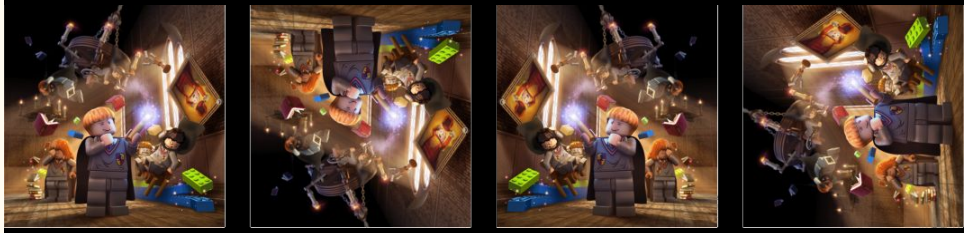
```
(define zipWith
   (lambda (f xs ys)
      (match (make-touple xs ys)
```

Schwierigkeiten A4 ÜB 8

```
(define groupby
  (lambda (group eq? xs)
    (map (lambda (a)
           (filter (lambda (b)
                     (eq? a (group b)))
                   xs))
         (distinct eq? (map group xs)))))
```

ÜB 9

Bilder spiegeln und Tixy



endrekursive Funktionen

$$f(x) = \begin{cases} s(x), & \text{falls } R(x) \\ f(r(x)), & \text{sonst} \end{cases}$$

Eine Funktion f ist endrekursiv wennder rekursive Funktionsaufruf die letzte Aktion zur Berechnung von f’ ist

```
sum(n):
  if n=0
     0
  else
     n + sum(n-1)
```

```
sum(n):
  return add_sum (0, n)

add_sum(m, n)
  if n=0
    m
  else
    add_sum (m+n, n-1)
```

```
sum(n):
  if n=0
      0
  else
      n + sum(n-1)
```

```
sum(3) = 3 + sum(2)
 sum(2) = 2 + sum(1)
  sum(1) = 1 + sum(0)
   sum(0) = 0
  sum(1) = 1 + 0 = 1
 sum(2) = 2 + 1 = 3
sum(3) = 3 + 3 = 6
```

```
sum(n):
 return add_sum (0, n)

add_sum(m, n)
 if n=0
  m
 else
  add_sum (m+n, n-1)
```

```
sum(3)   = add_sum(0, 3)
         = add_sum(3, 2)
         = add_sum(5, 1)
         = add_sum(6, 0)
         = 6
```

```
sum(n):
  if n=0
     0
  else
     n + sum(n-1)
```

nicht Endrekursiv

```
sum(n):
 return add_sum (0, n)

add_sum(m, n)
 if n=0
   m
 else
   add_sum (m+n, n-1)
```

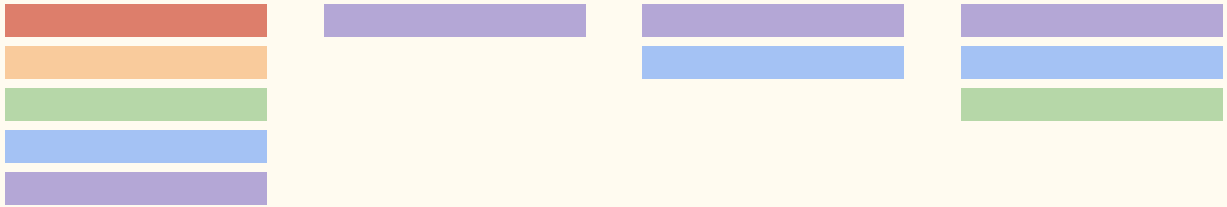
Endrekursiv

Bilder

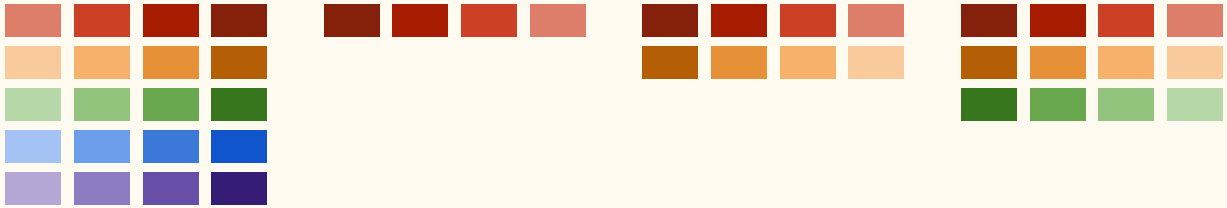
- image->color-list
  - Nimmt Bild, spuckt Liste an “pixeln” aus
- color-list->bitmap
    - Erzeugt Bild aus, “pixel”-Liste
- Drop
  - extrahiert die ersten n Elemente der Liste xs
- Take
  - entfernt die ersten n Elemente von xs

Bilder

Vertikale Spiegelung

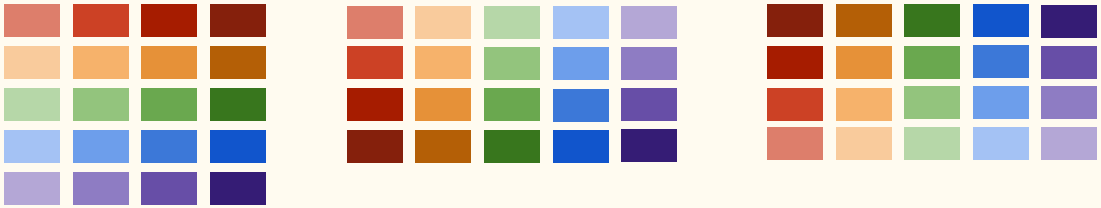


Horizontale Spiegelung



Bilder

90° Drehen



Wichtig: Die Teilaufgaben (b), (c) und (d) sollen unabhängig voneinander implementiert werden. Das bedeutet beispielsweise, dass die vertikale Spiegelung **nicht** mit Hilfe der horizontalen Spiegelung bzw. der Rotation implementiert werden soll.

Tixy

https://tixy.land/
https://forum-db.informatik.uni-tuebingen.de/t/der-grosse-tixy-thread-postet-eure-besten-tixies-hier/9055
https://doersino.github.io/tixyz/

Feedback: https://juleskreuer.eu/feedback