Listen

- Prozeduren für Listen:
  `rev, @, concat, map, filter, exists, all, length, hd, tl`
- Regelbasierte Funktionen:
```
fun length (x::xr) = 1 + length xs
  | length nil = 0
```
- Werfen von Ausnahmen:
```
fun hd nil = raise Empty
  | hd (x::xr) = x
```
- Faltung
```
foldl
foldr
```

Faltung

- `foldl:` $\forall \alpha, \beta \; . \; (\alpha * \beta \to \beta) \to \beta \to \alpha \; \mathit{list} \to \beta$

```
fun foldl f s nil = s
  | foldl f s (x::xr) = foldl f (f(x,s)) xr
```

- `foldr:` $\forall \alpha, \beta \; . \; (\alpha * \beta \to \beta) \to \beta \to \alpha \; \mathit{list} \to \beta$

```
fun foldr f s nil = s
  | foldr f s (x::xr) = f (x, foldr f s xr)
```

$\mathit{foldl} \; f \; s \; [x_1, x_2, x_3]$

$$\begin{tikzcd}
& f \arrow[dl] \arrow[dr] & & & \\
x_3 & & f \arrow[dl] \arrow[dr] & & \\
& x_2 & & f \arrow[dl] \arrow[dr] & \\
& & x_1 & & s
\end{tikzcd}$$

$\mathit{foldr} \; f \; s \; [x_1, x_2, x_3]$

$$\begin{tikzcd}
& f \arrow[dl] \arrow[dr] & & & \\
x_1 & & f \arrow[dl] \arrow[dr] & & \\
& x_2 & & f \arrow[dl] \arrow[dr] & \\
& & x_3 & & s
\end{tikzcd}$$

Beispiele

- `fun rev xs = foldl op::  nil xs`
- `fun append(xs, ys) = foldr op::  ys xs`
- `fun map f = foldr (fn (x,yr)=> (f x)::yr) nil`

-
  ```
  fun filter f = foldr
        (fn (x,ys) => if f x then x::ys else ys) nil
  ```

Split

```
fun split xs = foldl (fn (x,(xs,ys)) => (ys, x::xs))
(nil,nil) xs
                                            (nil, nil) xs

split [1,4,9,16]
= foldl f (nil,nil) [1,4,9,16]
= foldl f f(1,(nil,nil)) [4,9,19]
= foldl f f(4, (nil,[1])) [9,19]
= foldl f f(9, ([1],[4])) [19]
= foldl f f(19, ([4],[9,1]) []
= ([9,1], [19,4])
```