Reduce

Tijd voor meer zwarte magie

Som van een lijst

```
sum :: [Int] -> Int
sum []     = 0
sum (x:xs) = x + sum xs
```

Zijn alle elementen waar?

```
all :: [Bool] -> Bool
all []     = True
all (x:xs) = x && all xs
```

Som van een lijst

```
sum :: [Int] -> Int
sum []     = 0
sum (x:xs) = x + sum xs
```

Zijn alle elementen waar?

```
all :: [Bool] -> Bool
all []     = True
all (x:xs) = x && all xs
```

Abstractie

```
reduce :: (a -> a -> a) -> a -> [a] -> a
reduce f init []     = init
reduce f init (x:xs) = f x (reduce f init xs)
```

Abstractie

```
reduce :: (a -> a -> a) -> a -> [a] -> a
reduce f init []       = init
reduce f init (x:xs)   = f x (reduce f init xs)

sum :: [Int] -> Int
sum []       = 0
sum (x:xs) = x + sum xs

sum xs = reduce (+) 0 xs
```

Abstractie

```
reduce :: (a -> a -> a) -> a -> [a] -> a
reduce f init []       = init
reduce f init (x:xs)   = f x (reduce f init xs)

all :: [Bool] -> Bool
all []       = True
all (x:xs) = x && all xs

all xs = reduce (&&) True xs
```

Lijstconcatenatie

[[1,2],[3,4],[5,6,7]] -> [1,2,3,4,5,6,7]

Lijstconcatenatie

[[1,2],[3,4],[5,6,7]] -> [1,2,3,4,5,6,7]

```
concat xs = reduce (++) [] xs
```

Lijstconcatenatie

[[1,2],[3,4],[5,6,7]] -> [1,2,3,4,5,6,7]

```
concat xs = reduce (++) [] xs

reduce [] = []
reduce (x:xs) = x ++ reduce xs
```