Introduktion til Programmering og Problemløsning (PoP)

Produkt- og sumtyper

Records

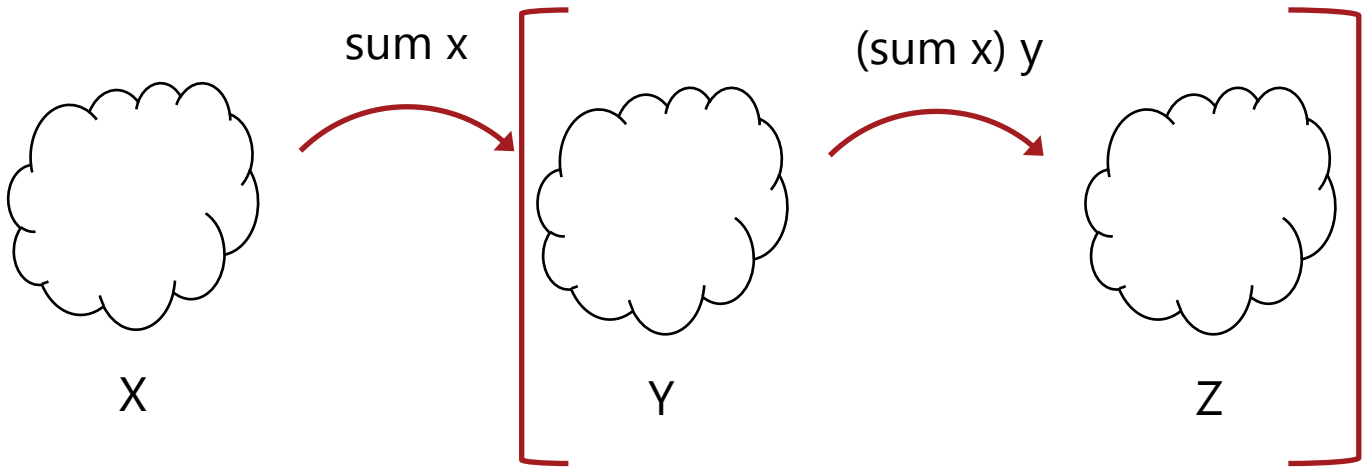
Jon Sporring
Department of Computer Science
2020/09/14

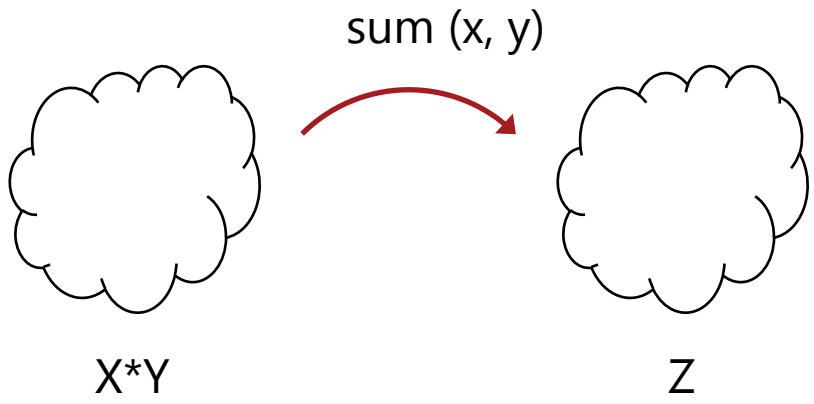
UNIVERSITY OF COPENHAGEN

Repetition: tuple argumenter

```
> let sum x y = x + y;;
val sum: x: int -> y: int -> int
> sum 3 4;;
val it: int = 7
```

```
> let sum (x, y) = x + y;;
val sum: x: int * y: int -> int
> sum (3, 4);;
val it: int = 7
```





Produkttype: Tupler

Parentes unødvendig men anbefalelses

Værdibinding

```
> let a = (1, 1.0);;
val a : int * float = (1, 1.0)
```

Produkttype

Mønstergenkendelse

```
> let (x,y) = a;;
val y: float = 1.0
val x: int = 1
```

Par-funktioner

```
> let x = fst a
- let y = snd a;;
val x: int = 1
val y: float = 1.0
```

Lav dine egen funktioner

```
let sum (p: float*float) : float =
  let (x,y) = p
  x + y;;
```

Alternativ notation

```
let sum (x: float, y: float) : float =
  x + y;;
```

Lav dine egene triple funktioner

```
let fst3 (x,_,_) = x
let snd3 (_,y,_) = y
let trd3 (_,_,z) = z
```

Argument ignoreres

Sumtyper (discriminative union): Semantik

Navne på Muligheder skal starte med stort bogstag

Eksempel: former

```
> type shape = Triangle | Square | Circle
- let toString (s:shape) : string =
-   match s with
-     Triangle -> "triangle"
-     | Square -> "square"
-     | Circle -> "circle"
- printfn "A Square string: %A" (toString Square)
A Square string: "square"
type shape =
  | Triangle
  | Square
  | Circle
val toString: s: shape -> string
val it: unit = ()
```

Alle tilfælde dækket, så wildcard er unødvendigt!

Sumtyper med data

Eksempel

Typesynonym (type abreviation)

Tuple af tupler

Husk kommentarer

Udpakning ved mønster genkendelse

Ignorér center

```
> type point = float*float
- type shape =
-   Rectangle of point*point // bottom left and top right
-   | Circle of point*float // center and radius
- let area (s:shape) : float =
-   match s with
-       Rectangle (lb, tr) ->
-       let (x1,y1) = lb
-       let (x2,y2) = tr
-       (x2-x1)*(y2-y1)
-     | Circle (_,r) ->
-       System.Math.PI*r*r
- let s = Rectangle ((1.0,1.0), (2.0,3.0))
- let c = Circle ((0.0,0.0), 3.5)
- printfn "The area of %A is %A " s (area s)
- printfn "The area of %A is %A " c (area c)
```

Interaktiv tilstand

```
The area of Rectangle ((1.0, 1.0), (2.0, 3.0)) is 2.0
The area of Circle ((0.0, 0.0), 3.5) is 38.48451001
type point = float * float
type shape =
  | Rectangle of point * point
  | Circle of point * float
val area: s: shape -> float
val s: shape = Rectangle ((1.0, 1.0), (2.0, 3.0))
val c: shape = Circle ((0.0, 0.0), 3.5)
val it: unit = ()
```

Optegnelser (records)

Record

```
type ansat = {
  id: int
  firstName: string
  lastName: string
  addresse: string
  email: string
}
let jon = {
  id = 2534
  firstName = "Jon"
  lastName = "Sporring"
  addresse = "Universitetsparken 1"
  email = "sporring@di.ku.dk"
}
```

Interaktiv tilstand

```
> printfn "Name: %A %A" jon.firstName jon.lastName;;
Name: "Jon" "Sporring"
val it: unit = ()
```

Optegnelser (records) og pattern matching

Record

```
type ansat = {
  id: int
  name: string
  email: string
}
let greetings (p: ansat) =
  match p with
    {id = 2534; name = "John"} -> "Error in data"
    | {id = 2534} -> "Greetings master of F#"
    | _ -> "Hello"

let jon = {
  id = 2534
  name = "Jon Sporring"
  email = "sporring@di.ku.dk"
}
printfn "%A" (greetings jon)
```

Interaktiv tilstand

```
"Greetings master of F#"
type ansat =
  {
    id: int
    name: string
    email: string
  }
val greetings: p: ansat -> string
val jon: ansat = { id = 2534
                   name = "John"
                   email =
"sporring@di.ku.dk" }
val it: unit = ()
```

Resumé

I denne video hørte du om:

- Produkttyper (tuples)
- Sumtyper (discriminated unions)
- Optegnelser (records)