OCAML

- functionele programmeertaal
- getypeerd
- O: object-oriented features
- ontwikkeld en gedistribueerd door INRIA (Frankrijk)

Voorbeeld: square

```
# let square (x) = x + x ;;

val square : int -> int = <fun>
```

Voorbeeld: insertion sort

```
# let rec sort = function
  | []     -> []
  | h :: t -> insert h (sort t)
  and
  insert elem = function
  | []     -> [elem]
  | h :: t -> if elem < h
                then elem :: h :: t
                else h :: insert elem t ;;
val sort : 'a list -> 'a list = <fun>
val insert : 'a -> 'a list -> 'a list = <fun>
```

Implementatie van λ-calculus

- door Freek Wiedijk
- `http://www.cs.kun.nl/~freek/notes/lambda.ml`
- in OCAML

Probleem

herbenoemen van variabelen

$$(\lambda x.\lambda y.x)y \rightarrow \lambda y'.\, y$$

Oplossing: De Bruijn-indexen

- geen namen voor variabelen maar natuurlijke getallen
- het getal geeft aan door welke lambda
  de variabele gebonden wordt
- $\lambda x.\, x$ is $\lambda\, 0$
  $\lambda xy.\, xy$ is $\lambda\lambda 1\, 0$

Probleem

$$(\lambda x.\lambda y.\, yx)\, z \rightarrow \lambda y.\, y\, z$$

naief met De Bruijn indexen:

$$(\lambda\lambda 0\, 1)0 \rightarrow \lambda 0\, 0$$

is fout !

Oplossing: lifting

nodig: liften van variabelen

$$(\lambda\lambda 0\,1)0 \to \lambda 0\,1$$

Leftmost-innermost

```
let rec leftmost_innermost t =
  match t with
    App(f,x) ->
     (try App(leftmost_innermost f,x)
      with Normal ->
        try App(f,leftmost_innermost x)
        with Normal ->
          beta id t)
  | Abstr(v,a) -> Abstr(v,leftmost_innermost a)
  | _ -> raise Normal;;
```

Leftmost-outermost

```
let rec leftmost_outermost t =
  match t with
    App(f,x) ->
     (try beta id t
      with Normal ->
        try App(leftmost_outermost f,x)
        with Normal ->
          App(f,leftmost_outermost x))
  | Abstr(v,a) -> Abstr(v,leftmost_outermost a)
  | _ -> raise Normal;;
```

Tentamenstof

- alles wat er op het college is behandeld

Termen

- termen:
  variabele (x), abstractie ($\lambda x.\, M$), applicatie ($F\, N$).

- termen als bomen

- haakjes:
  $\lambda x.\, x\, y$ is $\lambda x.\, (x\, y)$
  $x\, y\, z$ is $(x\, y)\, z$

Gebonden en vrije variabelen

- α-conversie:
  $\lambda x.\, xy$ is $\lambda z.\, zy$

Beta reductie

- β-reductieregel:
  $(\lambda x.\, M)\, N \rightarrow_\beta M[x := N]$
- substitutie !
- redex
- reductie, herschrijfrij
- normaalvorm
- geen δ-reductie

Datatypes

- Church numerals
  successor
  niet: Mul, Plus, Exp
- booleans
  not, and, or
- pairing
  projecties
- lijsten
  head, tail

Recursie

- niet: Y
- bewijs $\mathsf{Y}F = F(\mathsf{Y}F)$
- truuk

Terminatie

- een term P is terminerend
  als alle reductierijtjes beginnend in P eindig zijn

- Ω is niet terminerend

- sommige termen hebben een normaalvorm maar zijn ook het begin van een oneindige reductierij
  $(\lambda x.\, y)\Omega$

Reductiestrategieën

- een strategie schrijft voor hoe je een term moet reduceren
- doel: een normaalvorm vinden
  (of een weak head normal form)
- een strategie is normalizerend als er geldt: als een term een normaalvorm heeft dan vind je die normaalvorm door de strategie te volgen

Reductiestrategieën

- leftmost-innermost (call by value)
  (doel: normaalvorm)

- leftmost-outermost (call by need)
  (doel: normaalvorm)

- lazy reductiestategie
  (doel: weak head normal form)