

- DEV 8 -

Functioneel programmeren

...

Quick & Dirty Recap

Wat gaan we bespreken



- Lambda calculus
 - Expressies
 - Shadowing
 - Symbolen
 - Opdrachten

Wat gaan we bespreken

- F#
 - Code
 - Voorbeelden
 - Opdrachten



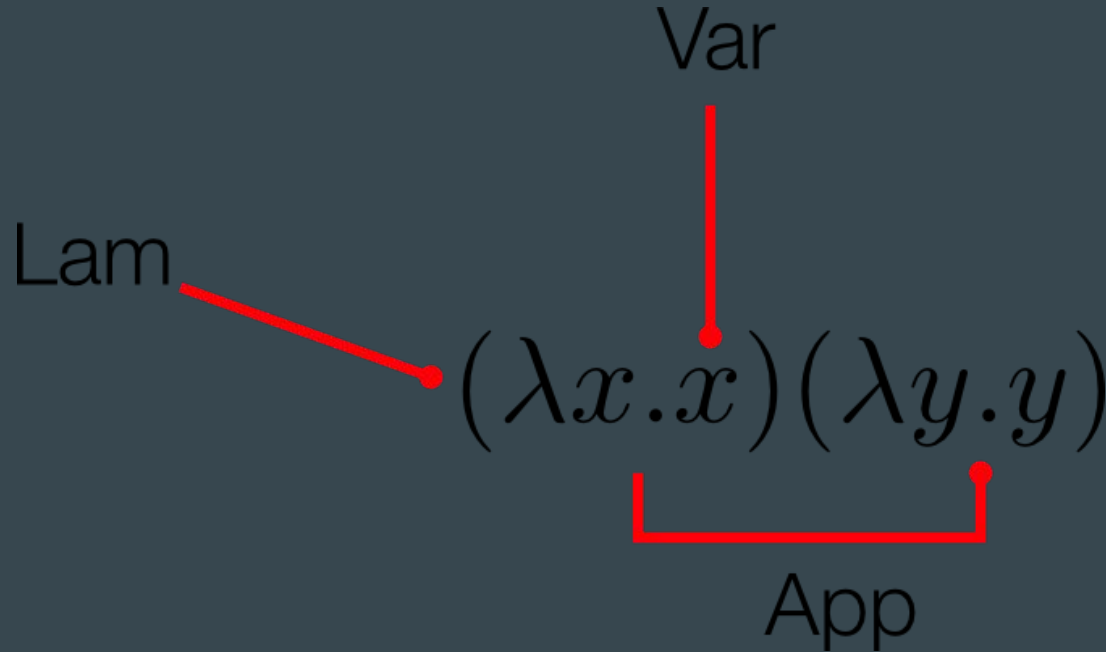
$\lambda x y . y \ 1 \ 0$

Lambda calculus

...

$\lambda x y . x$

Lambda calculus



Introductie - Expressies - Beta redux - Shadowing - Symbolen

The image features a dark blue-grey background with the text 'Lambda - Expressies' centered. Four sets of three parallel white lines are positioned in the corners: top-left, top-right, bottom-left, and bottom-right. Each set of lines is oriented diagonally, with the top-left and bottom-right sets sloping upwards and the top-right and bottom-left sets sloping downwards.

Lambda - Expressies

Lambda calculus



Lambda calculus is een “eenvoudige” taal.

Alle elementen in de taal zijn een “Expressie” (E).

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



Lambda calculus is een “eenvoudige” taal.

Mogelijke expressies (\mathcal{E}):

Variabele - x

Variabele - x

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



Lambda calculus is een “eenvoudige” taal.

Mogelijke expressies (E):

Variabele - x

Functie definitie - $\lambda E . E \parallel \lambda E \rightarrow E$

Variabele - x

Functie definitie - $\lambda x . x$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



Lambda calculus is een “eenvoudige” taal.

Mogelijke expressies (E):

Variabele - E

Functie definitie - $\lambda E . E \parallel \lambda E \rightarrow E$

Functie applicatie - $E E$

Variabele - x

Functie definitie - $\lambda x . x$

Functie applicatie - $(\lambda x . x) y$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



Variabele - E

Functie definitie - $\lambda E . E$

Functie applicatie - EE

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



Variabele - ϵ

Functie definitie - $\lambda \epsilon . \epsilon$

Functie applicatie - $\epsilon \epsilon$

```
def true(x, y):  
    return x
```

Definities:

(T) rue - $\lambda x y . x$

(F)alse - $\lambda x y . y$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



Variabele - ϵ

Functie definitie - $\lambda \epsilon . \epsilon$

Functie applicatie - $\epsilon \epsilon$

```
def false(x, y):  
    return y
```

Definities:

(T)true - $\lambda x y . x$

(F)alse - $\lambda x y . y$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



Variabele - ϵ

Functie definitie - $\lambda \epsilon . \epsilon$

Functie applicatie - $\epsilon \epsilon$

```
 $\lambda xy . x \ 1 \ 0$ 
```

```
 $((\lambda xy . x) \ 1) \ 0 \quad // (EE)E$ 
```

Definities:

(T)rue - $\lambda x \ y . x$

(F)alse - $\lambda x \ y . y$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

The image features a dark blue background with the text 'Lambda - Beta reduction' centered in white. The text is flanked by four sets of three parallel white lines, two in the top corners and two in the bottom corners, all slanted at approximately 45 degrees.

Lambda - Beta reduction

Lambda calculus



Definities:

(T)rue - $\lambda x y . x$

(F)alse - $\lambda x y . y$

1. $((T) 1) 0$

2. $((F) 1) 0$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



Definities:

(T)rue - $\lambda x y . x$

(F)alse - $\lambda x y . y$

(A)nd - $\lambda x y . x y x$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



Definities:

(T)rue - $\lambda x y . x$

(F)alse - $\lambda x y . y$

(A)nd - $\lambda x y . x y x$

```
def AND(x, y):  
    if x:  
        return y  
    return x
```

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus

Definities:

(T)true - $\lambda x y . x$

(F)false - $\lambda x y . y$

(A)nd - $\lambda x y . x y x$

$\lambda xy . xyx \quad F \quad T$

$((\lambda xy . xyx) \quad F) \quad T$

$(\lambda y . FyF) \quad T$

$(FTF) \quad // ((EE)E)$

F



Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



Definities:

(T)rue - $\lambda x y . x$

(F)alse - $\lambda x y . y$

(A)nd - $\lambda x y . x y x$

(O)r - $\lambda x y . x x y$

```
def OR(x, y):  
    if x:  
        return x  
    return y
```

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus

Definities:

(T)rue - $\lambda x y . x$

(F)alse - $\lambda x y . y$

(A)nd - $\lambda x y . x y x$

(O)r - $\lambda x y . x x y$

$\lambda xy . xxy \ F \ T$

$((\lambda xy . xxy) \ F) \ T$

$(\lambda y . FFy) \ T$

$(FFT) \ // ((EE)E)$

T



Introductie - Expressies - Beta redux - Shadowing - Symbolen

The image features a dark blue-grey background with the text 'Lambda - Shadowing' centered. In each of the four corners, there are three parallel white lines of varying lengths and orientations, creating a decorative frame. The lines in the top-left and bottom-right corners are oriented diagonally upwards, while the lines in the top-right and bottom-left corners are oriented diagonally downwards.

Lambda - Shadowing

Lambda calculus



Gebonden variabelen

$\lambda x y . x y$

Gebonden variabelen

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



Gebonden variabelen

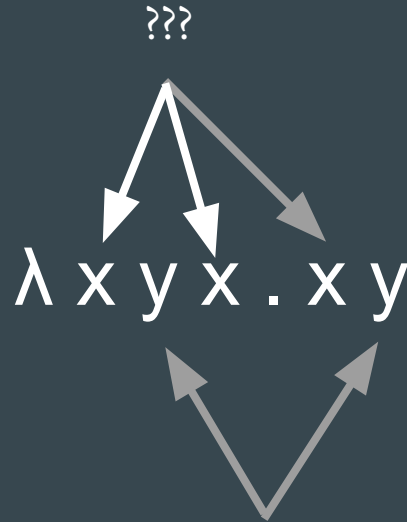
Vrije variabelen

$\lambda x y . x y z$

Gebonden variabelen

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



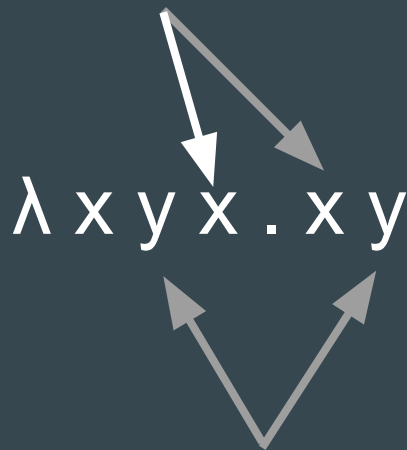
Gebonden variabelen

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



Gebonden variabelen



Gebonden variabelen

Introductie - Expressies - Beta redux - Shadowing - Symbolen

The image features a dark blue-grey background with the text "Lambda - symbolen" centered in white. The text is flanked by four sets of three parallel white lines, two in the top corners and two in the bottom corners, all slanted at approximately 45 degrees.

Lambda - symbolen

Lambda calculus



Wat is de volgende stap?

$$\overline{(\lambda x \rightarrow t) u \rightarrow_{\beta} t[x \mapsto u]}$$

$$\left\{ \begin{array}{l} x = \dots \\ t = \dots \\ u = \dots \\ t[x \mapsto u] = \dots \end{array} \right.$$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



$((\lambda y \ z. \ y) \ 1) \ 0$

$x =$

$t =$

$u =$

$t[\ x \rightarrow u] =$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



$((\lambda y \ z. \ y) \ 1) \ 0$

$x = y$

$t =$

$u =$

$t[\ x \rightarrow u] =$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



$((\lambda y \ z. \ y) \ 1) \ 0$

$x = y$

$t = (\lambda z. \ y)$

$u =$

$t[\ x \rightarrow u] =$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



$((\lambda y \ z. \ y) \ 1) \ 0$

$x = y$

$t = (\lambda z. \ y)$

$u = 1$

$t[\ x \rightarrow u] =$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



$((\lambda y \ z. \ y) \ 1) \ 0$

$x = y$

$t = (\lambda z. \ y)$

$u = 1$

$t[\ x \rightarrow u] = (\lambda z . \ 1) \ 0$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



$((\lambda y \ z. \ y) \ 1) \ 0$

$x = y$

$t = (\lambda z. \ y)$

$u = 1$

$t[\ x \rightarrow u] = (\lambda z . \ 1) \ 0$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



$((\lambda x y. x y) (\lambda z. z+1)) 5)$

$$\frac{t \rightarrow_{\beta} t' \wedge u \rightarrow u' \wedge t' u' \rightarrow_{\beta} v}{t u \rightarrow_{\beta} v}$$
$$\left\{ \begin{array}{l} t = \dots \\ u = \dots \\ t' = \dots \\ u' = \dots \\ v = \dots \end{array} \right.$$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



$(((\lambda x y. x y) (\lambda z. z+1)) 5)$

$t =$

$u =$

$t1 =$

$u1 =$

$v =$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



$(((\lambda x y. x y) (\lambda z. z+1)) 5)$

$t = (\lambda xy. x y) (\lambda z. z + 1)$

$u =$

$tl =$

$ul =$

$v =$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



$(((\lambda x y. x y) (\lambda z. z+1)) 5)$

$t = (\lambda xy. x y) (\lambda z. z + 1)$

$u = 5$

$t1 =$

$u1 =$

$v =$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



$(((\lambda x y. x y) (\lambda z. z+1)) 5)$

$t = (\lambda xy. x y) (\lambda z. z + 1)$

$u = 5$

$t1 = (\lambda y. (\lambda z. z + 1) y)$

$u1 =$

$v =$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



$(((\lambda x y. x y) (\lambda z. z+1)) 5)$

$t = (\lambda xy. x y) (\lambda z. z + 1)$

$u = 5$

$t1 = (\lambda y. (\lambda z. z + 1) y)$

$u1 = 5$

$v =$

Introductie - Expressies - Beta redux - Shadowing - Symbolen

Lambda calculus



$((\lambda x y. x y) (\lambda z. z+1)) 5$

$t = (\lambda x y. x y) (\lambda z. z + 1)$

$u = 5$

$t1 = (\lambda y. (\lambda z. z + 1) y)$

$u1 = 5$

$v = (\lambda z. z + 1) 5 \parallel 6$

Introductie - Expressies - Beta redux - Shadowing - Symbolen



Lambda calculus: Opdrachten

...

Lambda calculus - Opdrachten



1. $((\lambda xyz.zxy) \text{"Hi "}) \text{"Name "}) (\lambda xy. x + y + \text{"!"})$
2. $((\lambda xyz.y(zx)) 6) (\lambda y.y-4)) (\lambda x.x+2)$

Lambda calculus - Opdrachten



$((\lambda xyz.z \ x \ y) \ "Hi" \) \ "Name") \ (\lambda xy. \ x + y + "!")$

1. $((\lambda xyz.z \ x \ y) \ "Hi" \) \ "Name") \ (\lambda xy. \ x + y + "!")$
2. $((\lambda yz.z \ "Hi" \ y) \ "Name") \ (\lambda xy. \ x + y + "!")$
3. $(\lambda z.z \ "Hi" \ "Name") \ (\lambda xy. \ x + y + "!")$
4. $(\lambda xy. \ x + y + "!") \ "Hi" \ "Name"$
5. $(\lambda y. \ "Hi" \ + y + "!") \ "Name"$
6. $("Hi" \ + "Name" + "!")$
7. $"Hi \ Name!"$

Lambda calculus - Opdrachten



$((\lambda xyz.z \ x \ y) \ "Hi" \) \ "Name") \ (\lambda xy. \ x + y + "!")$

1. $((\lambda \underline{x}yz.z \ \underline{x} \ y) \ \underline{"Hi"} \) \ "Name") \ (\lambda xy. \ x + y + "!")$
2. $((\lambda \underline{y}z.z \ "Hi" \ \underline{y}) \ \underline{"Name"}) \ (\lambda xy. \ x + y + "!")$
3. $(\lambda \underline{z}. \underline{z} \ "Hi" \ "Name") \ (\lambda xy. \ x + y + "!")$
4. $(\lambda \underline{x}y. \ \underline{x} + y + "!") \ \underline{"Hi"} \ "Name"$
5. $(\lambda \underline{y}. \ "Hi" + \underline{y} + "!") \ \underline{"Name"}$
6. $(\underline{"Hi" + "Name" + "!"})$
7. $"Hi \ Name!"$

Lambda calculus - Opdrachten



$(((\lambda xyz.y (z x)) 6) (\lambda y.y-4)) (\lambda x.x+2)$

1. $(((\lambda xyz.y (z x)) 6) (\lambda y.y-4)) (\lambda x.x+2)$
2. $((\lambda yz.y (z 6)) (\lambda y.y-4)) (\lambda x.x+2)$
3. $(\lambda z.(\lambda y.y-4) (z 6)) (\lambda x.x+2)$
4. $(\lambda y.y-4) ((\lambda x.x+2) 6)$
5. $(\lambda y.y-4) (6+2)$
6. $6+2-4$
7. $8-4$
8. 4

Lambda calculus - Opdrachten



$((\lambda x y z. y (z x)) 6) (\lambda y. y-4)) (\lambda x. x+2)$

1. $((\lambda \underline{x} y z. y (z \underline{x})) \underline{6}) (\lambda y. y-4)) (\lambda x. x+2)$
2. $((\lambda \underline{y} z. \underline{y} (z 6)) \underline{(\lambda y. y-4)}) (\lambda x. x+2)$
3. $(\lambda \underline{z}. (\lambda y. y-4) (\underline{z} 6)) \underline{(\lambda x. x+2)}$
4. $(\lambda y. y-4) ((\lambda \underline{x}. \underline{x}+2) \underline{6})$
5. $(\lambda \underline{y}. \underline{y}-4) \underline{(6+2)}$
6. $\underline{6+2}-4$
7. $\underline{8-4}$
8. 4



F#

...

The image features a dark blue-grey background with the text 'F# - Syntax' centered in white. The corners of the image are decorated with sets of three parallel white lines. In the top-left and top-right corners, the lines are oriented diagonally upwards. In the bottom-left and bottom-right corners, the lines are oriented diagonally downwards.

F# - Syntax

F#



Slides en project

<https://github.com/CSARotterdam/Development-8-recap-1819>

01011000 01011000 01011000 01011000
01011000 01011000 01011000 01011000
01011000 01011000 01011000 01011000
01011000 01011000 01011000 01011000

Opdracht

...

01011000 01011000 01011000 01011000 01011000
01011000 01011000 01011000 01011000 01011000
01011000 01011000 01011000 01011000 01011000
01011000 01011000 01011000 01011000 01011000

F# - Opdrachten



Maak een power off functie. Het is alleen toegestaan om +, -, * en / operator te gebruiken. Functie definitie is als volgt:

```
n:int -> m:int -> int
```

F# - Opdrachten



If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.

Find the sum of all the multiples of 3 or 5 below 1000

F# - Opdrachten



En nu?

F# - Opdrachten



Opdrachten van:

Reader 1, 3 en 4

Vragen?

...



Feedback?

...

