

# Introduction to functional programming and lambda calculus

The INFDEV@HR Team

Hogeschool Rotterdam  
Rotterdam, Netherlands

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

# Introduction

## Lecture topics

- Course introduction
- Exam and practicum
- Semantics of traditional programming languages
- Basic lambda calculus

# Course introduction

## Course topics

- We will discuss a completely new paradigm for expressing programs
- This paradigm, functional programming, is based on different premises on computation
- It gives guarantees of correctness in complex places, like parallelism or separation of concerns
- It requires a radical conceptual shift in the way you think about programming

## Course topics

- We will begin with a short discussion on traditional programming language **semantics**
- We will then show the **lambda calculus**, which is the foundation for functional languages
- 
- We will then show how such language can be used to express mathematical concepts such as:
  - Defining sums between numbers
  - Defining multiplications between numbers
  - Defining divisions between numbers
  - Defining Logarithms
  - etc..

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

# Examination

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction  
Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

## Exam structure - idea

- There is a theoretical exam, where you show understanding of the basic principles
- There is a practical exam, where you show understanding of their concrete applications



Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

# Semantics of traditional programming languages

# Semantics of traditional programming languages

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

- Traditional, imperative programming languages are based on sharing memory through instructions
- This means that subsequent instructions are not independent from each other
- Any function call makes use of the available memory

# Semantics of traditional programming languages

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction  
Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

For example, consider the semantic rules that describe the working of “;”

First we run  $s_1$  with the initial memory, then we run  $s_2$  with the modified memory.

# Semantics of traditional programming languages

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction  
Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

For example, consider the semantic rules that describe the working of “;”

First we run  $s_1$  with the initial memory, then we run  $s_2$  with the modified memory.

$$\frac{\langle s_1, S, H \rangle \rightarrow \langle S_1, H_1 \rangle \wedge \langle s_2, S_1, H_1 \rangle \rightarrow \langle S_2, H_2 \rangle}{\langle (s_1; s_2), S, H \rangle \rightarrow \langle S_2, H_2 \rangle}$$

What does “*first we run  $s_1$  with the initial memory, then we run  $s_2$  with the modified memory*” imply?.

# Semantics of traditional programming languages

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

What does “*first we run  $s_1$  with the initial memory, then we run  $s_2$  with the modified memory*” imply?.

- The same instructions, executed at different moments, will produce **different results**.
- Change the order of some method calls, and some weird dependence might cause bugs or break things.

## Goals

- Our goal is to ensure that behaviour of code is consistent.
- Change the order of some method calls, and the results remain the same.
- This makes it easier to test, parallelize, and in general ensure correctness.

# Semantics of traditional programming languages

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

*How do we achieve this?*



*How do we achieve this?*

We give (shared) memory up: every piece of code is a function which output only depends on input.

This very important property is called **referential transparency**.

# Basic lambda calculus

## Introduction

- The (basic) lambda calculus is an alternative mechanism to Turing Machines and the Von Neumann architecture.
- It is very different, but has equivalent expressive power.
- It is the foundation of all functional programming languages.

## Substitution principle

- The (basic) lambda calculus is truly tiny when compared with its power.
- It is based on the substitution principle: calling a function with some parameters returns the function body with the variables replaced.
- There is no memory and no program counter: all we need to know is stored inside the body of the program itself.

A lambda calculus program (just *program* from now on) is made up of three syntactic elements:

- Variables:  $x, y, \dots$
- Abstractions (function declarations with one parameter):  $\lambda x \rightarrow t$  where  $x$  is a variable and  $t$  is the function body (a program).
- Applications (function calls with one argument):  $t u$  where  $t$  is the function being called (a program) and  $u$  is its argument (another program).

# Basic lambda calculus

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

A simple example would be the identity function, which just returns whatever it gets as input

$$(\lambda x \rightarrow x)$$

We can call this function with a variable as argument, by writing:

$$((\lambda x \rightarrow x) \ v)$$

# Basic lambda calculus

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

A lambda calculus program is computed by replacing lambda abstractions applied to arguments with the body of the lambda abstraction with the argument instead of the lambda parameter:



# Basic lambda calculus

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

A lambda calculus program is computed by replacing lambda abstractions applied to arguments with the body of the lambda abstraction with the argument instead of the lambda parameter:

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

$t[x \mapsto u]$  means that we change variable  $x$  with  $u$  within  $t$

# Basic lambda calculus

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda x \rightarrow x) \ v)$$

# Basic lambda calculus

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda x \rightarrow x) \ v)$$
$$((\lambda x \rightarrow x) \ v)$$

# Basic lambda calculus

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda x \rightarrow x) \ v)$$

# Basic lambda calculus

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda x \rightarrow x) \ v)$$
$$v$$

# Basic lambda calculus

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

Multiple applications where the left-side is not a lambda abstraction are solved in a left-to-right fashion:

# Basic lambda calculus

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

Multiple applications where the left-side is not a lambda abstraction are solved in a left-to-right fashion:

$$\frac{t \rightarrow_{\beta} t' \wedge u \rightarrow_{\beta} u' \wedge t' u' \rightarrow_{\beta} v}{t u \rightarrow_{\beta} v}$$

# Basic lambda calculus

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

Variables cannot be further reduced, that is they stay the same:



# Basic lambda calculus

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

Variables cannot be further reduced, that is they stay the same:

$$\overline{x \rightarrow_{\beta} x}$$

We can encode functions with multiple parameters by nesting lambda abstractions:

$$(\lambda x \ y \rightarrow (x \ y))$$

The parameters are then given one at a time:

$$(((\lambda x \ y \rightarrow (x \ y)) \ A) \ B)$$

# Basic lambda calculus

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$(((\lambda x \ y \rightarrow (x \ y)) \ A) \ B)$$

# Basic lambda calculus

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$(((\lambda x \ y \rightarrow (x \ y)) \ A) \ B)$$
$$( ((\lambda x \ y \rightarrow (x \ y)) \ A) \ B)$$

# Basic lambda calculus

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda x. y \rightarrow (x \ y)) \ A) \ B$$

# Basic lambda calculus

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda x. y \rightarrow (x \ y)) \ A) \ B$$
$$((\lambda y. \rightarrow (A \ y)) \ B)$$

# Basic lambda calculus

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda y \rightarrow (A \ y)) \ B)$$



# Basic lambda calculus

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda y \rightarrow (A \ y)) \ B)$$
$$((\lambda y \rightarrow (A \ y)) \ B)$$

# Basic lambda calculus

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda y \rightarrow (A \ y)) \ B)$$

# Basic lambda calculus

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda y \rightarrow (A \ y)) \ B)$$
$$(A \ B)$$

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

# Closing up

## Example executions of (apparently) nonsensical programs

- We will now exercise with the execution of various lambda programs.
- Try to guess what the result of these programs is, and then we shall see what would have happened.

*What is the result of this program execution?*

$$(((\lambda x \ y \rightarrow (x \ y)) \ (\lambda z \rightarrow (z \ z))) \ A)$$

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$(((\lambda x \ y \rightarrow (x \ y)) \ (\lambda z \rightarrow (z \ z))) \ A)$$

$$(((\lambda x \ y \rightarrow (x \ y)) \ (\lambda z \rightarrow (z \ z))) \ A)$$
$$( \ ((\lambda x \ y \rightarrow (x \ y)) \ (\lambda z \rightarrow (z \ z))) \ A)$$



Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda x. y \rightarrow (x \ y)) (\lambda z. \rightarrow (z \ z))) \ A$$

$$((\lambda x. y \rightarrow (x \ y)) (\lambda z. \rightarrow (z \ z))) \ A)$$
$$((\lambda y. \rightarrow ((\lambda z. \rightarrow (z \ z)) \ y)) \ A)$$

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda y \rightarrow ((\lambda z \rightarrow (z \ z)) \ y)) \ A)$$

$$((\lambda y \rightarrow ((\lambda z \rightarrow (z \ z)) \ y)) \ A)$$
$$((\lambda y \rightarrow ((\lambda z \rightarrow (z \ z)) \ y)) \ A)$$

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda y \rightarrow ((\lambda z \rightarrow (z \ z)) \ y)) \ A)$$

$$((\lambda y \rightarrow ((\lambda z \rightarrow (z \ z)) \ y)) \ A)$$
$$((\lambda z \rightarrow (z \ z)) \ A)$$

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda z \rightarrow (z \ z)) \ A)$$

$$((\lambda z \rightarrow (z \ z)) \ A)$$
$$((\lambda z \rightarrow (z \ z)) \ A)$$



Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda z \rightarrow (z \ z)) \ A)$$

# Closing up

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda z \rightarrow (z \ z)) \ A)$$
$$( \ A \ A )$$

*What is the result of this program execution? Watch out for the scope of the two “x” variables!*

```
(( (λx x → (x x)) A) B)
```

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$(((\lambda x \ x \rightarrow (x \ x)) \ A) \ B)$$

$$(((\lambda x \ x \rightarrow (x \ x)) \ A) \ B)$$
$$( ((\lambda x \ x \rightarrow (x \ x)) \ A) \ B)$$

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda x. x \rightarrow (x \ x)) \ A) \ B$$

$$((\lambda x. x \rightarrow (x \ x)) \ A) \ B$$
$$((\lambda x. x \rightarrow (x \ x)) \ B)$$

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda x \rightarrow (x \ x)) \ B)$$



Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda x \rightarrow (x \ x)) \ B)$$
$$((\lambda x \rightarrow (x \ x)) \ B)$$

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda x \rightarrow (x \ x)) \ B)$$

$$((\lambda x \rightarrow (x \ x)) \ B)$$
$$( \ B \ B )$$

The first “x” gets replaced with “A”, but the second “x” shadows it!

$$(((\lambda x \ x \rightarrow (x \ x)) \ A) \ B)$$

A better formulation, less ambiguous, would turn:

$$(((\lambda x \ x \rightarrow (x \ x)) \ A) \ B)$$

...into:

$$(((\lambda y \ x \rightarrow (x \ x)) \ A) \ B)$$

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$(((\lambda y \ x \rightarrow (x \ x)) \ A) \ B)$$

$$(((\lambda y \ x \rightarrow (x \ x)) \ A) \ B)$$
$$( ((\lambda y \ x \rightarrow (x \ x)) \ A) \ B)$$

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda y \ x \rightarrow (x \ x)) \ A) \ B$$



$$((\lambda y \ x \rightarrow (x \ x)) \ A) \ B)$$
$$((\lambda x \rightarrow (x \ x)) \ B)$$

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda x \rightarrow (x \ x)) \ B)$$

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda x \rightarrow (x \ x)) \ B)$$
$$((\lambda x \rightarrow (x \ x)) \ B)$$

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda x \rightarrow (x \ x)) \ B)$$

$$((\lambda x \rightarrow (x \ x)) \ B)$$
$$( \ B \ B )$$

*What is the result of this program execution? Is there even a result?*

$$((\lambda x \rightarrow (x \ x)) \ (\lambda x \rightarrow (x \ x)))$$

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda x \rightarrow (x \ x)) \ (\lambda x \rightarrow (x \ x)))$$

$$((\lambda x \rightarrow (x \ x)) \ (\lambda x \rightarrow (x \ x)))$$
$$((\lambda x \rightarrow (x \ x)) \ (\lambda x \rightarrow (x \ x)))$$



Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda x \rightarrow (x \ x)) (\lambda x \rightarrow (x \ x)))$$

$$((\lambda x \rightarrow (x \ x)) (\lambda x \rightarrow (x \ x)))$$
$$((\lambda x \rightarrow (x \ x)) (\lambda x \rightarrow (x \ x)))$$

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda x \rightarrow (x \ x)) \ (\lambda x \rightarrow (x \ x)))$$

$$((\lambda x \rightarrow (x \ x)) \ (\lambda x \rightarrow (x \ x)))$$
$$((\lambda x \rightarrow (x \ x)) \ (\lambda x \rightarrow (x \ x)))$$

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

$$((\lambda x \rightarrow (x \ x)) (\lambda x \rightarrow (x \ x)))$$

$$((\lambda x \rightarrow (x \ x)) (\lambda x \rightarrow (x \ x)))$$
$$((\lambda x \rightarrow (x \ x)) (\lambda x \rightarrow (x \ x)))$$

$$((\lambda x \rightarrow (x \ x)) \ (\lambda x \rightarrow (x \ x)))$$

It never ends! Like a while true: ..!

Ok, I know what you are all thinking: what is this for sick joke?  
This is no real programming language!

- We have some sort of functions and function calls
- We do not have booleans and if's
- We do not have integers and arithmetic operators
- We do not have a lot of things!



## Surprise!

With nothing but lambda programs we will show how to build all of these features and more.

Introduction  
to functional  
programming  
and lambda  
calculus

The  
INFDEV@HR  
Team

Introduction

Course  
introduction

Examination

Semantics of  
traditional  
programming  
languages

Basic lambda  
calculus

Closing up

Stay tuned.

This will be a marvelous voyage.

The best of luck, and thanks for the  
attention!