

Functional Programming for Logicians

Péter Mekis

Department of Logic, ELTE Budapest

Session 1: 2019 February 11

Programming styles

Functional

vs

Procedural

- Computation is evaluation of mathematical functions.

Programming styles

Functional

vs

Procedural

- Computation is evaluation of mathematical functions.

- Computation is execution of procedures.

Programming styles

Functional

vs

Procedural

- Computation is evaluation of mathematical functions.
- A function transforms data values to data values.

- Computation is execution of procedures.

Programming styles

Functional

vs

Procedural

- Computation is evaluation of mathematical functions.
- A function transforms data values to data values.

- Computation is execution of procedures.
- A procedure is a sequence of commands, executed sequentially, transforming program states to program states.

Functional

vs

Procedural

- Computation is evaluation of mathematical functions.
- A function transforms data values to data values.
- A program is a network of function definitions with function calls inside the definitions.

- Computation is execution of procedures.
- A procedure is a sequence of commands, executed sequentially, transforming program states to program states.

Functional

vs

Procedural

- Computation is evaluation of mathematical functions.
- A function transforms data values to data values.
- A program is a network of function definitions with function calls inside the definitions.

- Computation is execution of procedures.
- A procedure is a sequence of commands, executed sequentially, transforming program states to program states.
- A program is a network of procedure definitions, with procedure calls inside the definitions.

Typed Lambda Calculus

- Primitive types (eg. Int for integers, Char for characters, Bool for Booleans)

Typed Lambda Calculus

- Primitive types (eg. Int for integers, Char for characters, Bool for Booleans)
- Function types: $\alpha \rightarrow \beta$ is a function that takes a value of type α , and returns a value of type β .

Typed Lambda Calculus

- Primitive types (eg. Int for integers, Char for characters, Bool for Booleans)
- Function types: $\alpha \rightarrow \beta$ is a function that takes a value of type α , and returns a value of type β .
- Function application: $f_{\alpha \rightarrow \beta} g_{\alpha}$ has type β .

Typed Lambda Calculus

- Primitive types (eg. Int for integers, Char for characters, Bool for Booleans)
- Function types: $\alpha \rightarrow \beta$ is a function that takes a value of type α , and returns a value of type β .
- Function application: $f_{\alpha \rightarrow \beta} g_{\alpha}$ has type β .
- Lambda abstraction: $(\lambda. x_{\alpha} h_{\beta})$ has type $\alpha \rightarrow \beta$

Typed Lambda Calculus

- Primitive types (eg. Int for integers, Char for characters, Bool for Booleans)
- Function types: $\alpha \rightarrow \beta$ is a function that takes a value of type α , and returns a value of type β .
- Function application: $f_{\alpha \rightarrow \beta} g_{\alpha}$ has type β .
- Lambda abstraction: $(\lambda. x_{\alpha} h_{\beta})$ has type $\alpha \rightarrow \beta$
- Binary function types: $\alpha \rightarrow (\beta \rightarrow \gamma)$ takes a value of type α , and returns a function that takes a value of type β , and returns a value of type γ .

Examples for various types

- $\text{Int} \rightarrow \text{Int}$: unary operations on integers

Examples for various types

- $\text{Int} \rightarrow \text{Int}$: unary operations on integers
eg. $(+3) 5 = 8$; $\text{succ } 12 = 13$

Examples for various types

- $\text{Int} \rightarrow \text{Int}$: unary operations on integers
eg. $(+3) 5 = 8$; $\text{succ } 12 = 13$
- $\text{Char} \rightarrow \text{Char}$: unary operations on characters

Examples for various types

- $\text{Int} \rightarrow \text{Int}$: unary operations on integers
eg. $(+3) \ 5 = 8$; $\text{succ} \ 12 = 13$
- $\text{Char} \rightarrow \text{Char}$: unary operations on characters
eg. $\text{nextchar} \ 'c' = 'd'$, $\text{upper} \ 'a' = 'A'$

Examples for various types

- $\text{Int} \rightarrow \text{Int}$: unary operations on integers
eg. $(+3) 5 = 8$; $\text{succ } 12 = 13$
- $\text{Char} \rightarrow \text{Char}$: unary operations on characters
eg. $\text{nextchar } 'c' = 'd'$, $\text{upper } 'a' = 'A'$
- $\text{Int} \rightarrow \text{Bool}$: properties of integers

Examples for various types

- $\text{Int} \rightarrow \text{Int}$: unary operations on integers
eg. $(+3) 5 = 8$; $\text{succ } 12 = 13$
- $\text{Char} \rightarrow \text{Char}$: unary operations on characters
eg. $\text{nextchar } 'c' = 'd'$, $\text{upper } 'a' = 'A'$
- $\text{Int} \rightarrow \text{Bool}$: properties of integers
eg. $\text{even } 7 = \text{False}$, $\text{prime } 7 = \text{True}$

Examples for various types

- $\text{Int} \rightarrow \text{Int}$: unary operations on integers
eg. $(+3) \ 5 = 8$; $\text{succ } 12 = 13$
- $\text{Char} \rightarrow \text{Char}$: unary operations on characters
eg. $\text{nextchar } 'c' = 'd'$, $\text{upper } 'a' = 'A'$
- $\text{Int} \rightarrow \text{Bool}$: properties of integers
eg. $\text{even } 7 = \text{False}$, $\text{prime } 7 = \text{True}$
- $\text{Char} \rightarrow \text{Bool}$: properties of characters

Examples for various types

- $\text{Int} \rightarrow \text{Int}$: unary operations on integers
eg. $(+3) \ 5 = 8$; $\text{succ } 12 = 13$
- $\text{Char} \rightarrow \text{Char}$: unary operations on characters
eg. $\text{nextchar } 'c' = 'd'$, $\text{upper } 'a' = 'A'$
- $\text{Int} \rightarrow \text{Bool}$: properties of integers
eg. $\text{even } 7 = \text{False}$, $\text{prime } 7 = \text{True}$
- $\text{Char} \rightarrow \text{Bool}$: properties of characters
eg. $\text{vowel } 'a' = \text{True}$, $\text{isupper } 's' = \text{False}$

Examples for various types

- $\text{Int} \rightarrow \text{Int}$: unary operations on integers
eg. $(+3) 5 = 8$; $\text{succ } 12 = 13$
- $\text{Char} \rightarrow \text{Char}$: unary operations on characters
eg. $\text{nextchar } 'c' = 'd'$, $\text{upper } 'a' = 'A'$
- $\text{Int} \rightarrow \text{Bool}$: properties of integers
eg. $\text{even } 7 = \text{False}$, $\text{prime } 7 = \text{True}$
- $\text{Char} \rightarrow \text{Bool}$: properties of characters
eg. $\text{vowel } 'a' = \text{True}$, $\text{isupper } 's' = \text{False}$
- $\text{Bool} \rightarrow \text{Bool}$: properties of truth values

Examples for various types

- $\text{Int} \rightarrow \text{Int}$: unary operations on integers
eg. $(+3) \ 5 = 8$; $\text{succ } 12 = 13$
- $\text{Char} \rightarrow \text{Char}$: unary operations on characters
eg. $\text{nextchar } 'c' = 'd'$, $\text{upper } 'a' = 'A'$
- $\text{Int} \rightarrow \text{Bool}$: properties of integers
eg. $\text{even } 7 = \text{False}$, $\text{prime } 7 = \text{True}$
- $\text{Char} \rightarrow \text{Bool}$: properties of characters
eg. $\text{vowel } 'a' = \text{True}$, $\text{isupper } 's' = \text{False}$
- $\text{Bool} \rightarrow \text{Bool}$: properties of truth values
eg. $\text{isfalse } 'True' = \text{False}$ (is this function familiar...?)

Examples for various types

- $\text{Int} \rightarrow \text{Int}$: unary operations on integers
eg. $(+3) \ 5 = 8$; $\text{succ} \ 12 = 13$
- $\text{Char} \rightarrow \text{Char}$: unary operations on characters
eg. $\text{nextchar} \ 'c' = 'd'$, $\text{upper} \ 'a' = 'A'$
- $\text{Int} \rightarrow \text{Bool}$: properties of integers
eg. $\text{even} \ 7 = \text{False}$, $\text{prime} \ 7 = \text{True}$
- $\text{Char} \rightarrow \text{Bool}$: properties of characters
eg. $\text{vowel} \ 'a' = \text{True}$, $\text{isupper} \ 's' = \text{False}$
- $\text{Bool} \rightarrow \text{Bool}$: properties of truth values
eg. $\text{isfalse} \ 'True' = \text{False}$ (is this function familiar...?)
- $\text{Int} \rightarrow (\text{Int} \rightarrow \text{Int})$: binary operations on integers

Examples for various types

- $\text{Int} \rightarrow \text{Int}$: unary operations on integers
eg. $(+3) \ 5 = 8$; $\text{succ} \ 12 = 13$
- $\text{Char} \rightarrow \text{Char}$: unary operations on characters
eg. $\text{nextchar} \ 'c' = 'd'$, $\text{upper} \ 'a' = 'A'$
- $\text{Int} \rightarrow \text{Bool}$: properties of integers
eg. $\text{even} \ 7 = \text{False}$, $\text{prime} \ 7 = \text{True}$
- $\text{Char} \rightarrow \text{Bool}$: properties of characters
eg. $\text{vowel} \ 'a' = \text{True}$, $\text{isupper} \ 's' = \text{False}$
- $\text{Bool} \rightarrow \text{Bool}$: properties of truth values
eg. $\text{isfalse} \ 'True' = \text{False}$ (is this function familiar...?)
- $\text{Int} \rightarrow (\text{Int} \rightarrow \text{Int})$: binary operations on integers
eg. $\text{squaresum} \ 5 \ 7 = 74$; $\text{min} \ 8 \ 12 = 8$

Examples for various types

- $\text{Int} \rightarrow \text{Int}$: unary operations on integers
eg. $(+3) \ 5 = 8$; $\text{succ } 12 = 13$
- $\text{Char} \rightarrow \text{Char}$: unary operations on characters
eg. $\text{nextchar } 'c' = 'd'$, $\text{upper } 'a' = 'A'$
- $\text{Int} \rightarrow \text{Bool}$: properties of integers
eg. $\text{even } 7 = \text{False}$, $\text{prime } 7 = \text{True}$
- $\text{Char} \rightarrow \text{Bool}$: properties of characters
eg. $\text{vowel } 'a' = \text{True}$, $\text{isupper } 's' = \text{False}$
- $\text{Bool} \rightarrow \text{Bool}$: properties of truth values
eg. $\text{isfalse } 'True' = \text{False}$ (is this function familiar...?)
- $\text{Int} \rightarrow (\text{Int} \rightarrow \text{Int})$: binary operations on integers
eg. $\text{squaresum } 5 \ 7 = 74$; $\text{min } 8 \ 12 = 8$
- $\text{Int} \rightarrow (\text{Int} \rightarrow \text{Bool})$: relations of integers

Examples for various types

- $\text{Int} \rightarrow \text{Int}$: unary operations on integers
eg. $(+3) \ 5 = 8$; $\text{succ } 12 = 13$
- $\text{Char} \rightarrow \text{Char}$: unary operations on characters
eg. $\text{nextchar } 'c' = 'd'$, $\text{upper } 'a' = 'A'$
- $\text{Int} \rightarrow \text{Bool}$: properties of integers
eg. $\text{even } 7 = \text{False}$, $\text{prime } 7 = \text{True}$
- $\text{Char} \rightarrow \text{Bool}$: properties of characters
eg. $\text{vowel } 'a' = \text{True}$, $\text{isupper } 's' = \text{False}$
- $\text{Bool} \rightarrow \text{Bool}$: properties of truth values
eg. $\text{isfalse } 'True' = \text{False}$ (is this function familiar...?)
- $\text{Int} \rightarrow (\text{Int} \rightarrow \text{Int})$: binary operations on integers
eg. $\text{squaresum } 5 \ 7 = 74$; $\text{min } 8 \ 12 = 8$
- $\text{Int} \rightarrow (\text{Int} \rightarrow \text{Bool})$: relations of integers
eg. $\text{twinprimes } 11 \ 13 = \text{True}$; $(<) \ 7 \ 5 = \text{False}$

- Type declarations are part of the object level syntax.

- Type declarations are part of the object level syntax.
- Type variables range over types, and are subject to lambda abstraction.

- Type declarations are part of the object level syntax.
- Type variables range over types, and are subject to lambda abstraction.
- We can define type classes: e.g. $\text{Prop} = (\alpha \rightarrow \text{Bool})$

- Type declarations are part of the object level syntax.
- Type variables range over types, and are subject to lambda abstraction.
- We can define type classes: e.g. $\text{Prop} = (\alpha \rightarrow \text{Bool})$
- $\text{Int} \rightarrow \text{Bool}$, $\text{Char} \rightarrow \text{Bool}$, and $\text{Bool} \rightarrow \text{Bool}$ are all instances of the Prop class.

Haskell's type system: a first glance

- Int: 32 bit integers from -2^{29} to $2^{29} - 1$.

Haskell's type system: a first glance

- Int: 32 bit integers from -2^{29} to $2^{29} - 1$.
- Integer: integers of arbitrary size

Haskell's type system: a first glance

- Int: 32 bit integers from -2^{29} to $2^{29} - 1$.
- Integer: integers of arbitrary size
- Char: characters

Haskell's type system: a first glance

- Int: 32 bit integers from -2^{29} to $2^{29} - 1$.
- Integer: integers of arbitrary size
- Char: characters
- Bool: truth values

Haskell's type system: a first glance

- Int: 32 bit integers from -2^{29} to $2^{29} - 1$.
- Integer: integers of arbitrary size
- Char: characters
- Bool: truth values
- $a \rightarrow b$: unary functions mapping from type a to type b

Haskell's type system: a first glance

- Int: 32 bit integers from -2^{29} to $2^{29} - 1$.
- Integer: integers of arbitrary size
- Char: characters
- Bool: truth values
- $a \rightarrow b$: unary functions mapping from type a to type b
- $a \rightarrow (a \rightarrow b)$: binary functions mapping from types a and b to type c
(Parentheses are usually omitted.)

Haskell's type system: a first glance

- `[a]`: the type class of lists

Haskell's type system: a first glance

- `[a]`: the type class of lists
- Lists are homogeneous, and have arbitrary length

Haskell's type system: a first glance

- `[a]`: the type class of lists
- Lists are homogeneous, and have arbitrary length
- `[Int]` is the type of 32 bit integer lists; an instance of `[a]`.

Haskell's type system: a first glance

- `[a]`: the type class of lists
- Lists are homogeneous, and have arbitrary length
- `[Int]` is the type of 32 bit integer lists; an instance of `[a]`.
eg. `[0,1,2,3,4]`; `[5,6,7]`

Haskell's type system: a first glance

- `[a]`: the type class of lists
- Lists are homogeneous, and have arbitrary length
- `[Int]` is the type of 32 bit integer lists; an instance of `[a]`.
eg. `[0,1,2,3,4]`; `[5,6,7]`
- `String = [Char]`, strings are also instances of `[a]`.

Haskell's type system: a first glance

- `[a]`: the type class of lists
- Lists are homogeneous, and have arbitrary length
- `[Int]` is the type of 32 bit integer lists; an instance of `[a]`.
eg. `[0,1,2,3,4]`; `[5,6,7]`
- `String = [Char]`, strings are also instances of `[a]`.
eg. `"abc"`, same as `['a','b','c']`,

Haskell's type system: a first glance

- `[a]`: the type class of lists
- Lists are homogeneous, and have arbitrary length
- `[Int]` is the type of 32 bit integer lists; an instance of `[a]`.
eg. `[0,1,2,3,4]`; `[5,6,7]`
- `String = [Char]`, strings are also instances of `[a]`.
eg. `"abc"`, same as `['a','b','c']`,
- `[[a]]` the type class of list of lists; a subclass of `[a]`

Haskell's type system: a first glance

- `[a]`: the type class of lists
- Lists are homogeneous, and have arbitrary length
- `[Int]` is the type of 32 bit integer lists; an instance of `[a]`.
eg. `[0,1,2,3,4]`; `[5,6,7]`
- `String = [Char]`, strings are also instances of `[a]`.
eg. `"abc"`, same as `['a','b','c']`,
- `[[a]]` the type class of list of lists; a subclass of `[a]`
eg. `[[], [0,1], [3]]`; `["D. Trump", "B. Obama", "G. W. Bush"]`

Haskell's type system: a first glance

- `[a]`: the type class of lists
- Lists are homogeneous, and have arbitrary length
- `[Int]` is the type of 32 bit integer lists; an instance of `[a]`.
eg. `[0,1,2,3,4]`; `[5,6,7]`
- `String = [Char]`, strings are also instances of `[a]`.
eg. `"abc"`, same as `['a','b','c']`,
- `[[a]]` the type class of list of lists; a subclass of `[a]`
eg. `[[], [0,1], [3]]`; `["D. Trump", "B. Obama", "G. W. Bush"]`
- `[a] -> a` is the type class of functions from lists to elements

Haskell's type system: a first glance

- `[a]`: the type class of lists
- Lists are homogeneous, and have arbitrary length
- `[Int]` is the type of 32 bit integer lists; an instance of `[a]`.
eg. `[0,1,2,3,4]`; `[5,6,7]`
- `String = [Char]`, strings are also instances of `[a]`.
eg. `"abc"`, same as `['a','b','c']`,
- `[[a]]` the type class of list of lists; a subclass of `[a]`
eg. `[[], [0,1], [3]]`; `["D. Trump", "B. Obama", "G. W. Bush"]`
- `[a] -> a` is the type class of functions from lists to elements
eg. `head`

Haskell's type system: a first glance

- `[a]`: the type class of lists
- Lists are homogeneous, and have arbitrary length
- `[Int]` is the type of 32 bit integer lists; an instance of `[a]`.
eg. `[0,1,2,3,4]`; `[5,6,7]`
- `String = [Char]`, strings are also instances of `[a]`.
eg. `"abc"`, same as `['a','b','c']`,
- `[[a]]` the type class of list of lists; a subclass of `[a]`
eg. `[[], [0,1], [3]]`; `["D. Trump", "B. Obama", "G. W. Bush"]`
- `[a] -> a` is the type class of functions from lists to elements
eg. `head`
- `[a -> b]` is the type class of lists of unary functions

Haskell's type system: a first glance

- `[a]`: the type class of lists
- Lists are homogeneous, and have arbitrary length
- `[Int]` is the type of 32 bit integer lists; an instance of `[a]`.
eg. `[0,1,2,3,4]`; `[5,6,7]`
- `String = [Char]`, strings are also instances of `[a]`.
eg. `"abc"`, same as `['a','b','c']`,
- `[[a]]` the type class of list of lists; a subclass of `[a]`
eg. `[[], [0,1], [3]]`; `["D. Trump", "B. Obama", "G. W. Bush"]`
- `[a] -> a` is the type class of functions from lists to elements
eg. `head`
- `[a -> b]` is the type class of lists of unary functions
eg. `even`, `odd`, `prime`