Introduction to Lambda Calculus
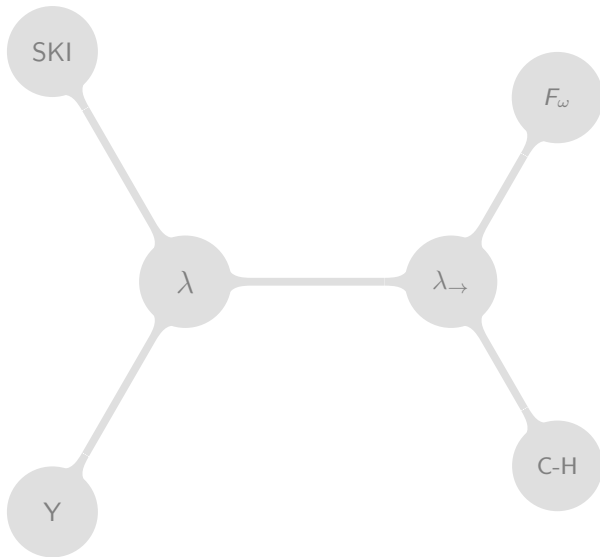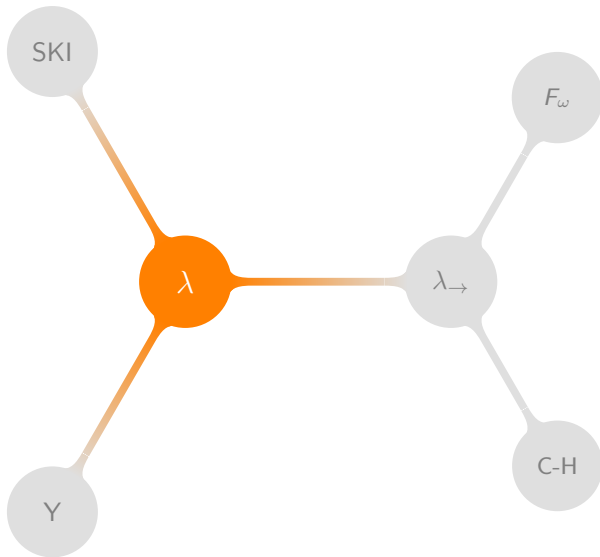
Maciek Makowski (@mmakowski)

12th November 2014

The Plan

Basic Lambda Calculus

Semantics

$$f(x) = 3 * x + 2$$

Semantics

$$f(x) = 3 * x + 2$$
$$\lambda x. + (* 3\, x)\, 2$$

Syntax

$$\langle term \rangle ::= x \qquad\qquad\qquad\qquad \text{(variable)}$$
$$| \quad (\lambda x.\langle term \rangle) \qquad\qquad \text{(abstraction)}$$
$$| \quad (\langle term \rangle \ \langle term \rangle) \qquad \text{(application)}$$

where $x \in \mathbb{X}$ – the set of variables

Syntax

v_1

Syntax

v_1 var v_1

Syntax

$x\ y$

Syntax

$x\ y$

```
         app
        /   \
    var x   var y
```
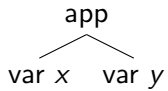
Syntax
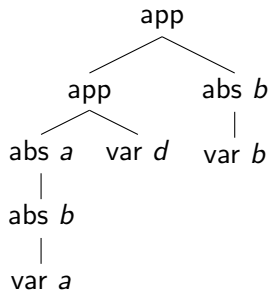
$\lambda a.b$

Syntax

$\lambda a.b$

```
abs a
  |
var b
```

Syntax

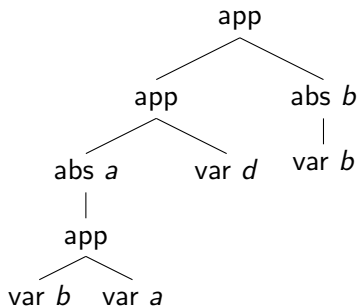$(\lambda a.\lambda b.a)\ d\ (\lambda b.b)$

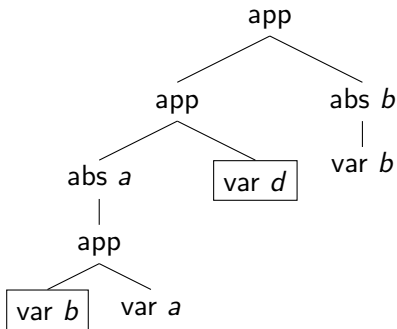# Syntax

$(\lambda a.\lambda b.a)\ d\ (\lambda b.b)$

Syntax

$(\lambda a.b\ a)\ d\ (\lambda b.b)$

Syntax

$(\lambda a.\underline{b}\ a)\ \underline{d}\ (\lambda b.b)$

Syntax

- terms: trees consisting of
  - variables
  - abstractions
  - applications
- variables are *bound* by abstraction; otherwise *free*

Rewriting
α-conversion

$$(\lambda x.x\, y)\, (\lambda x.x) \longleftrightarrow_\alpha (\lambda a.a\, y)\, (\lambda b.b)$$

Rewriting
β-reduction

$$(\lambda x.M)\ N \longrightarrow_\beta M[x/N]$$

Rewriting
β-reduction

$$(\lambda x.M)\ N \longrightarrow_\beta M[x/N]$$

$$(\lambda x.x\,y)\ (\lambda z.z) \longrightarrow_\beta (\lambda z.z)\ y \longrightarrow_\beta y$$

Rewriting

β-reduction

$$(\lambda x.M)\ N \longrightarrow_\beta M[x/N]$$

$$(\lambda a.a\ (\lambda a.a))\ b \longrightarrow_\beta b\ (\lambda a.a)$$
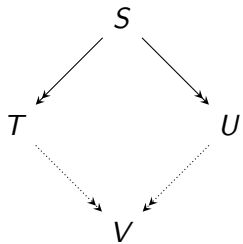
Rewriting
β-reduction

$$(\lambda x.M) \ N \longrightarrow_\beta M[x/N]$$

TODO: nicer redexes

$(\lambda a.(\lambda b.b)[1]a)[3]((\lambda c.\lambda d.d)[2]\lambda f.f)$

Rewriting
Church-Rosser

$$
\begin{array}{ccc}
 & S & \\
\swarrow & & \searrow \\
T & & U \\
\searrow & & \swarrow \\
 & V &
\end{array}
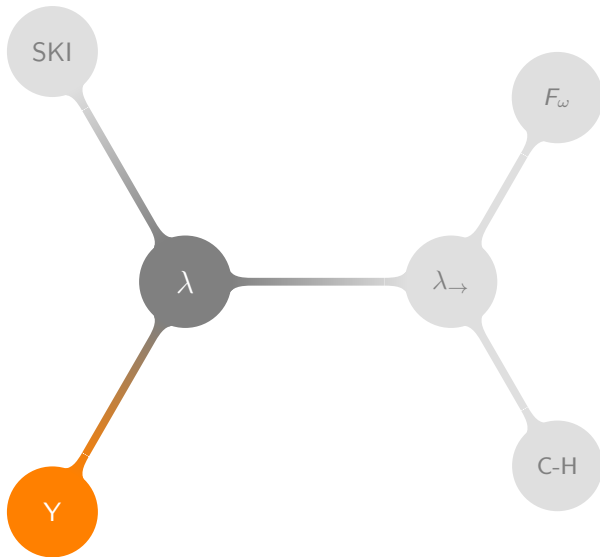$$

Rewriting

β-reduction

$$\Omega = (\lambda x.xx)(\lambda x.xx)$$

Programming in Lambda Calculus

Conditionals

if C then T else F

Conditionals

if *C* then *T* else *F*

$$\texttt{true} = \lambda t.\lambda f.t$$
$$\texttt{false} = \lambda t.\lambda f.f$$

Conditionals

if C then T else F

$$\text{true} = \lambda t.\lambda f.t$$
$$\text{false} = \lambda t.\lambda f.f$$
$$\text{test} = \lambda c.\lambda t.\lambda f.c\, t\, f$$
$$\text{if } C \text{ then } T \text{ else } F = \text{test } C\, T\, F$$

Numbers

$$0 = \lambda s.\lambda z.z$$
$$\mathtt{succ} = \lambda n.\lambda s.\lambda z.s\,(n\,s\,z)$$

Numbers

$$0 = \lambda s.\lambda z.z$$
$$\texttt{succ} = \lambda n.\lambda s.\lambda z.s\,(n\,s\,z)$$

$$0 = \qquad\qquad \lambda s.\lambda z.z$$
$$1 = \texttt{succ}\ 0 = \lambda s.\lambda z.s\,z$$
$$2 = \texttt{succ}\ 1 = \lambda s.\lambda z.s\,(s\,z)$$
$$3 = \texttt{succ}\ 2 = \lambda s.\lambda z.s\,(s\,(s\,z))$$
$$\vdots$$
$$\texttt{n} = \qquad\qquad \lambda s.\lambda z\,\underbrace{s\,(\ldots s\,(s\,z)\ldots)}_{n}$$

Numbers

$$0 = \lambda s.\lambda z.z$$
$$\texttt{succ} = \lambda n.\lambda s.\lambda z.s\,(n\,s\,z)$$

$$\texttt{plus} = \lambda m.\lambda n.\lambda s.\lambda z.m\,s\,(n\,s\,z)$$
$$\texttt{times} = \lambda m.\lambda n.m\,(\texttt{plus}\,n)\,0$$

Recursion

$$n! = \begin{cases} 1 & \text{if } n = 0, \\ n * (n-1)! & \text{otherwise.} \end{cases}$$

Recursion

$$\mathtt{Y} = \lambda f.(\lambda x.f\,(x\,x))\,(\lambda x.f\,(x\,x))$$

Recursion

$$Y = \lambda f.(\lambda x.f\,(x\,x))\,(\lambda x.f\,(x\,x))$$

$$g = \lambda f.\lambda n.\texttt{if eq } n \text{ 0 then 1 else } (\texttt{times n}\,(f\,(\texttt{pred n})))$$

$$\texttt{factorial} = Y\,g$$

Recursion

$$Y = \lambda f.(\lambda x.f\,(x\,x))\,(\lambda x.f\,(x\,x))$$
$$g = \lambda f.\lambda n.\texttt{if eq } n\ 0 \texttt{ then 1 else }(\texttt{times n}\,(f\,(\texttt{pred n})))$$
$$\texttt{factorial} = Y\,g$$

```
factorial 3
Y g 3
(h h) 3                    where h = λx.g (x x)
g (h h) 3
g fct 3                    where fct = h h
if eq 3 0 then 1 else (times 3 (fct (pred 3)))
```
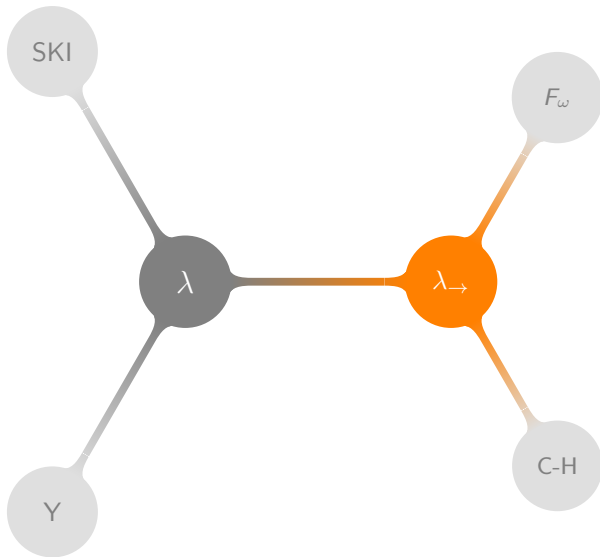
Simple Types

Simple Types

$$\langle type \rangle ::= \sigma$$
$$\qquad | \quad (\langle type \rangle \rightarrow \langle type \rangle)$$
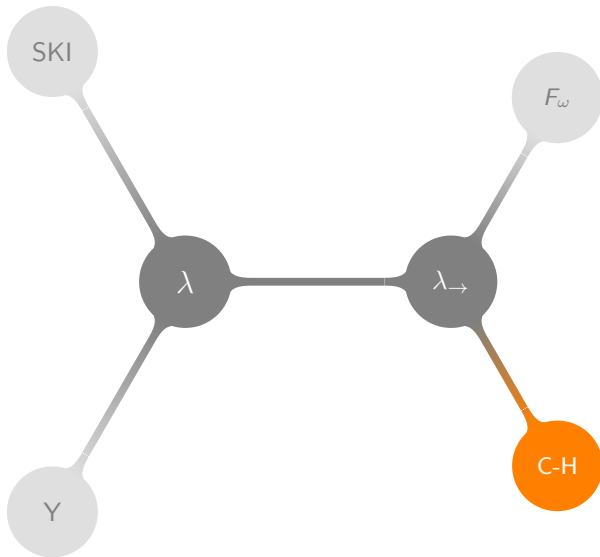
Simple Types

$$\frac{M : \sigma \to \tau \qquad N : \sigma}{M\,N : \tau} \qquad \text{(application)}$$

$$\frac{\begin{array}{c} \cancel{x : \sigma} \\ \vdots \\ M : \tau \end{array}}{\lambda x.M : \sigma \to \tau} \qquad \text{(abstraction)}$$

Curry-Howard Correspondence

Curry-Howard Correspondence

$$\frac{\phi \Rightarrow \psi \qquad \phi}{\psi} \qquad \text{(implication elimination)}$$

$$\frac{\begin{array}{c} \cancel{\phi} \\ \vdots \\ \psi \end{array}}{\phi \Rightarrow \psi} \qquad \text{(implication introduction)}$$

Curry-Howard Correspondence

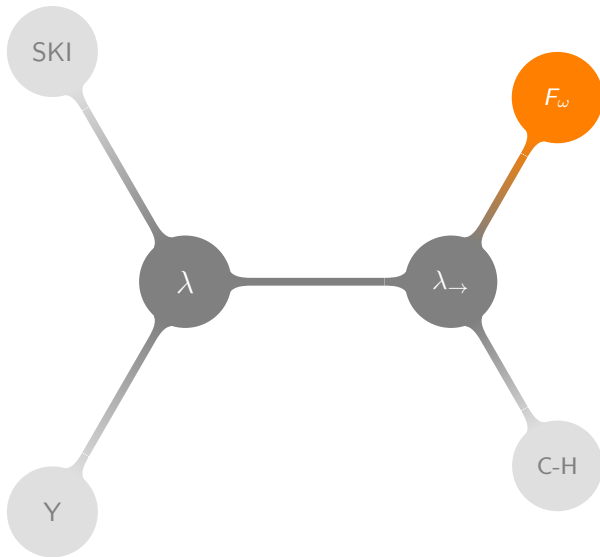$$\frac{\phi \Rightarrow \psi \qquad \phi}{\psi} \qquad\qquad \frac{M : \sigma \to \tau \qquad N : \sigma}{M\,N : \tau}$$
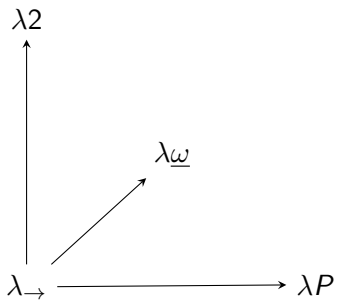
$$\frac{\begin{array}{c} \not\phi \\ \vdots \\ \psi \end{array}}{\phi \Rightarrow \psi} \qquad\qquad \frac{\begin{array}{c} \cancel{x : \sigma} \\ \vdots \\ M : \tau \end{array}}{\lambda x.M : \sigma \to \tau}$$
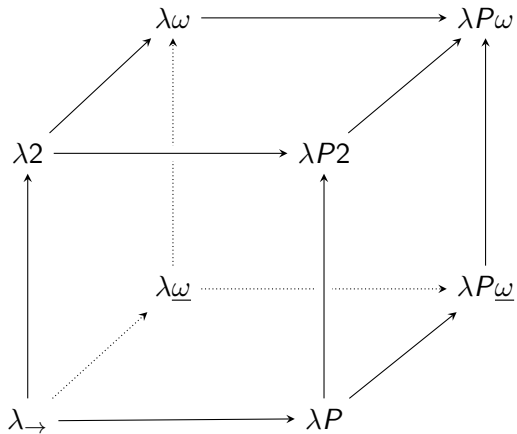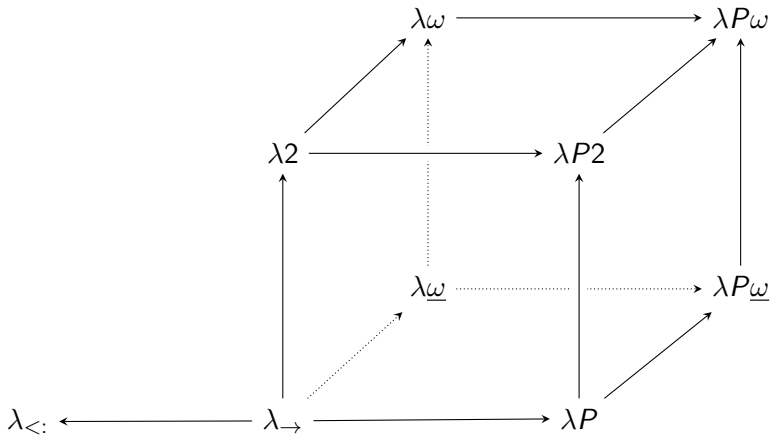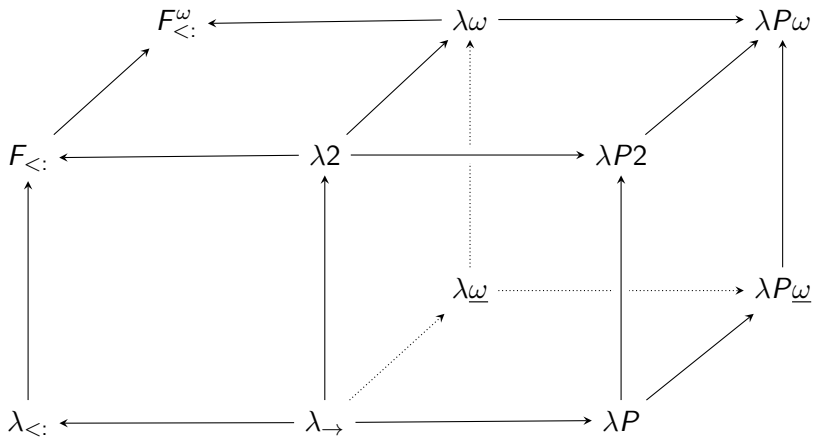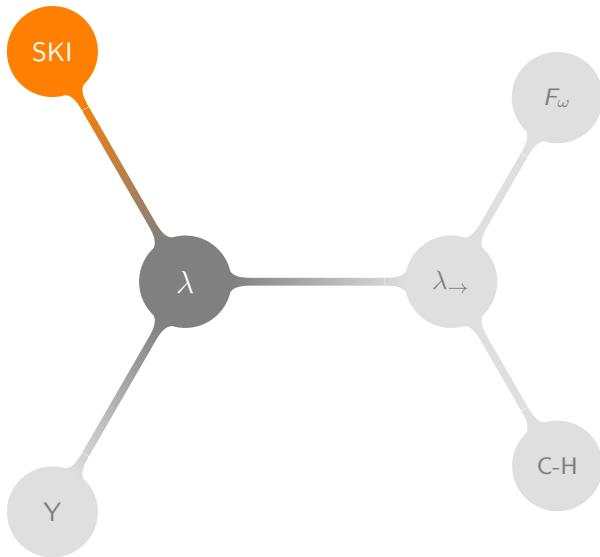
More Types

The Lambda Cube

The Lambda Cube

Subtyping

Subtyping

Combinatory Logic

Combinatory Logic

$$\mathrm{K} = \lambda x.\lambda y.x$$
$$\mathrm{S} = \lambda x.\lambda y.\lambda z.xz(yz)$$
$$\mathrm{I} = \mathrm{S}\,\mathrm{K}\,\mathrm{K}$$

Combinatory Logic

$$K = \lambda x.\lambda y.x$$
$$S = \lambda x.\lambda y.\lambda z.xz(yz)$$
$$I = S\,K\,K$$

$$\lambda x.\lambda y.yx = S\,(K\,(S\,I))\,(S\,(K\,K)\,I)$$

Combinatory Logic

$$\mathtt{X} = \lambda x. x\, \mathtt{S}\, \mathtt{K}$$

Combinatory Logic

$$\mathtt{X} = \lambda x. x\, \mathtt{S}\, \mathtt{K}$$

$$\mathtt{K} = \mathtt{X}\,(\mathtt{X}\,(\mathtt{X}\,\mathtt{X}))$$
$$\mathtt{S} = \mathtt{X}\,(\mathtt{X}\,(\mathtt{X}\,(\mathtt{X}\,\mathtt{X})))$$

Further Reading

- Benjamin C. Pierce, *Types and Programming Languages*
- Morten Heine B. Sørensen, Paweł Urzyczyn, *Lectures on the Curry-Howard Isomorphism*
- Henk Berendregt, Erik Barendsen, *Introduction to Lambda Calculus*
- Henk Berendregt *The Lambda Calculus, its Syntax and Semantics*

Notes

- https://github.com/mmakowski/introlambda/blob/baml/notes.pdf
- https://github.com/mmakowski/introlambda/blob/baml/slides.pdf