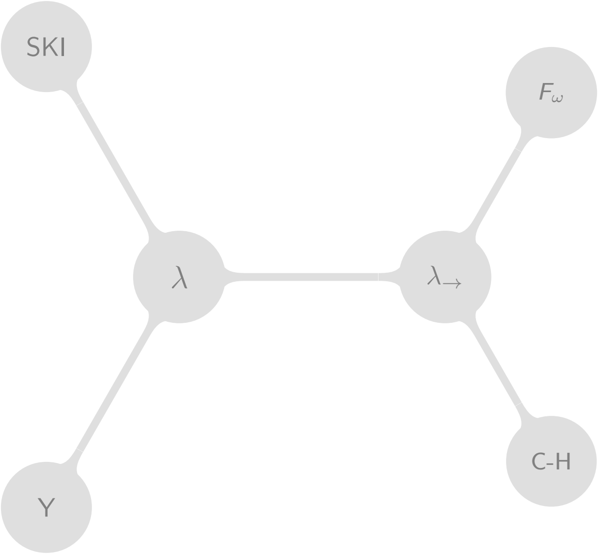
Introduction to Lambda Calculus

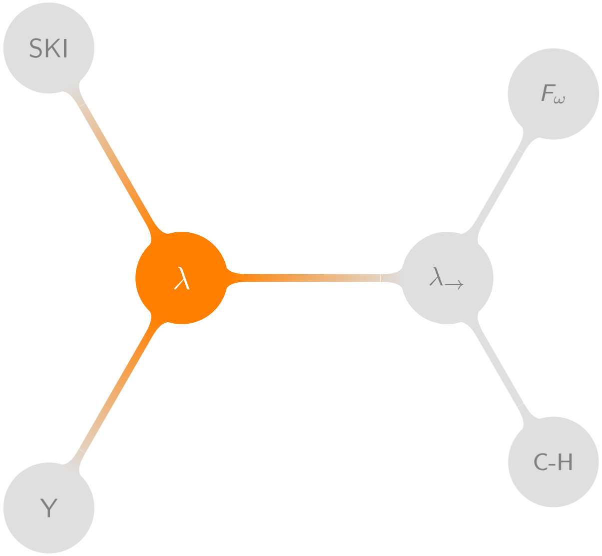
Maciek Makowski (@mmakowski)

1st October 2014

The Plan



Basic Lambda Calculus



Syntax

$$
\begin{array}{rcll}
\langle \textit{term} \rangle & ::= & x & \text{(variable)} \\
& | & (\lambda x.\langle \textit{term} \rangle) & \text{(abstraction)} \\
& | & (\langle \textit{term} \rangle\ \langle \textit{term} \rangle) & \text{(application)}
\end{array}
$$

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

Syntax

v_1

Syntax

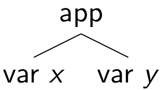
v_1 var v_1

Syntax

$x \; y$

Syntax

$x\ y$



Syntax

$\lambda a.b$

Syntax

$\lambda a.b$

$$\begin{array}{c} \mathsf{abs}\ a \\ | \\ \mathsf{var}\ b \end{array}$$

Syntax

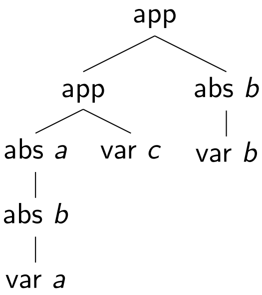
$$(\lambda a.\lambda b.a)\ c\ (\lambda b.b)$$

Syntax

$$
\begin{array}{rcll}
\langle \mathit{term} \rangle & ::= & x & \text{(variable)} \\
& \mid & (\lambda x.\langle \mathit{term} \rangle) & \text{(abstraction)} \\
& \mid & (\langle \mathit{term} \rangle\ \langle \mathit{term} \rangle) & \text{(application)}
\end{array}
$$

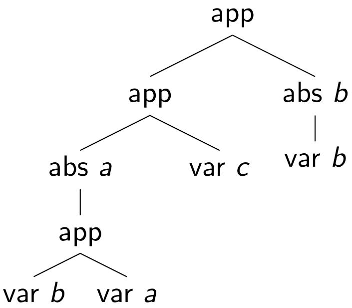
Syntax

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



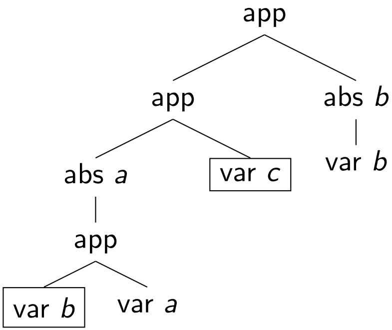
Syntax

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



Syntax

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



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

- *call-by-value*: start with innermost redex, do not reduce under abstraction
- *call-by-name*: start with outermost redex, do not reduce under abstraction

Rewriting

Church-Rosser

$$
\begin{tikzcd}
 & S \arrow[dl] \arrow[dr] & \\
T \arrow[dr, dotted] & & U \arrow[dl, dotted] \\
 & V &
\end{tikzcd}
$$

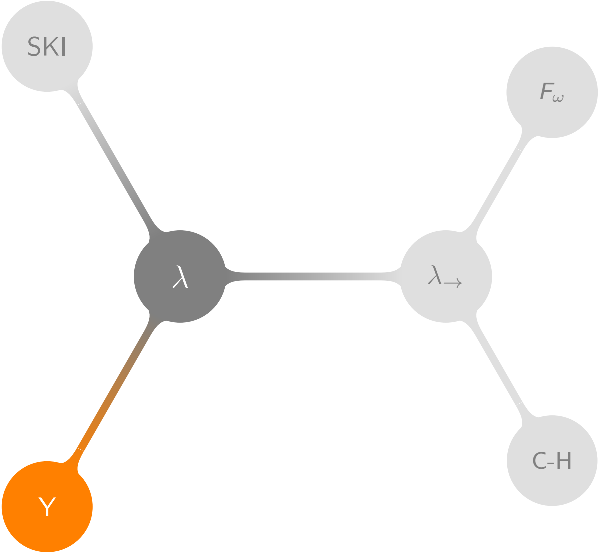
Rewriting

TODO: Ω

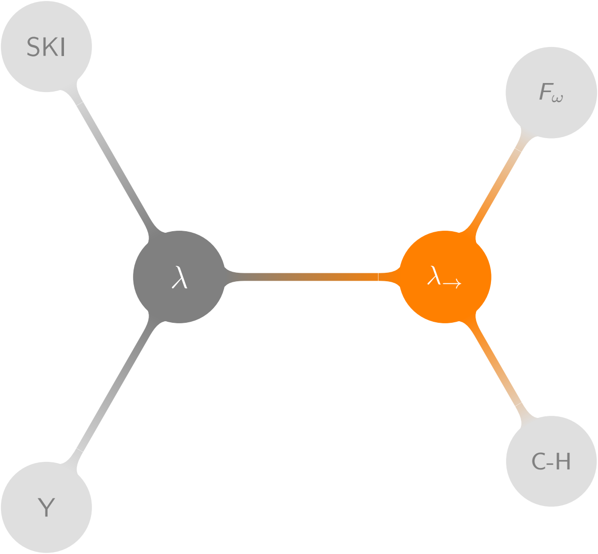
Semantics

TODO: functions

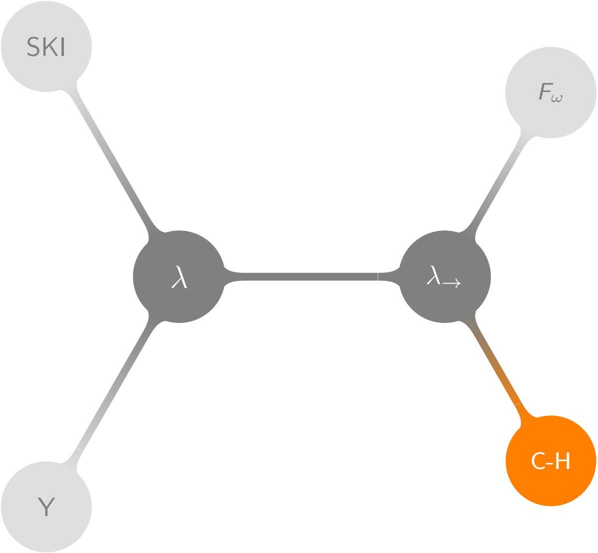
Programming in Lambda Calculus



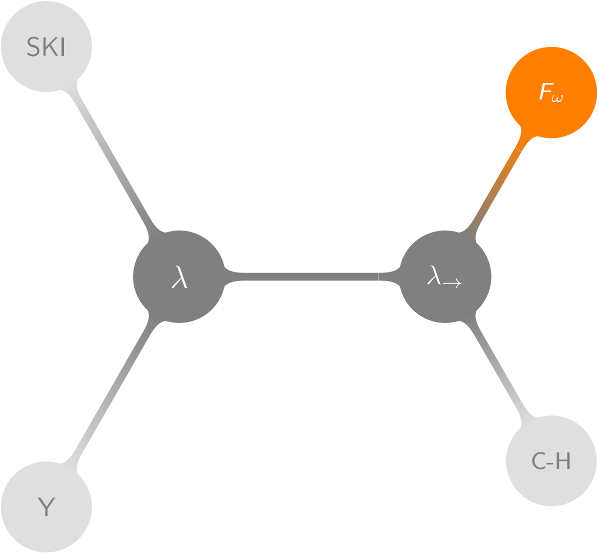
Simple Types



Curry-Howard Correspondence



More Types



The Lambda Cube

$$
\begin{tikzcd}
\lambda 2 & & \\
& \lambda\underline{\omega} & \\
\lambda_{\to} \arrow[uu] \arrow[ur] \arrow[rr] & & \lambda P
\end{tikzcd}
$$

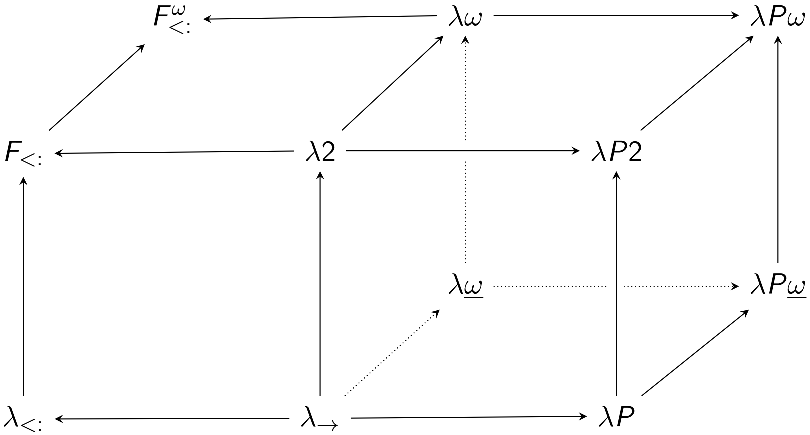
The Lambda Cube

$$
\begin{tikzcd}
 & \lambda\omega \arrow[rr] & & \lambda P\omega \\
\lambda 2 \arrow[ur] \arrow[rr] & & \lambda P2 \arrow[ur] & \\
 & \lambda\underline{\omega} \arrow[uu, dotted] \arrow[rr, dotted] & & \lambda P\underline{\omega} \arrow[uu] \\
\lambda_{\to} \arrow[uu] \arrow[ur, dotted] \arrow[rr] & & \lambda P \arrow[uu] \arrow[ur] &
\end{tikzcd}
$$

Subtyping

$$
\begin{tikzcd}
 & & \lambda\omega \arrow[rr] & & \lambda P\omega \\
 & \lambda 2 \arrow[ur] \arrow[rr] & & \lambda P2 \arrow[ur] & \\
 & & \lambda\underline{\omega} \arrow[uu, dotted] \arrow[rr, dotted] & & \lambda P\underline{\omega} \arrow[uu] \\
\lambda_{<:} & \lambda_{\to} \arrow[l] \arrow[uu] \arrow[ur, dotted] \arrow[rr] & & \lambda P \arrow[uu] \arrow[ur] &
\end{tikzcd}
$$

Subtyping


$F^{\omega}_{<:}$
$\lambda\omega$
$\lambda P\omega$
$F_{<:}$
$\lambda 2$
$\lambda P2$
$\lambda\underline{\omega}$
$\lambda P\underline{\omega}$
$\lambda_{<:}$
$\lambda_{\to}$
λP