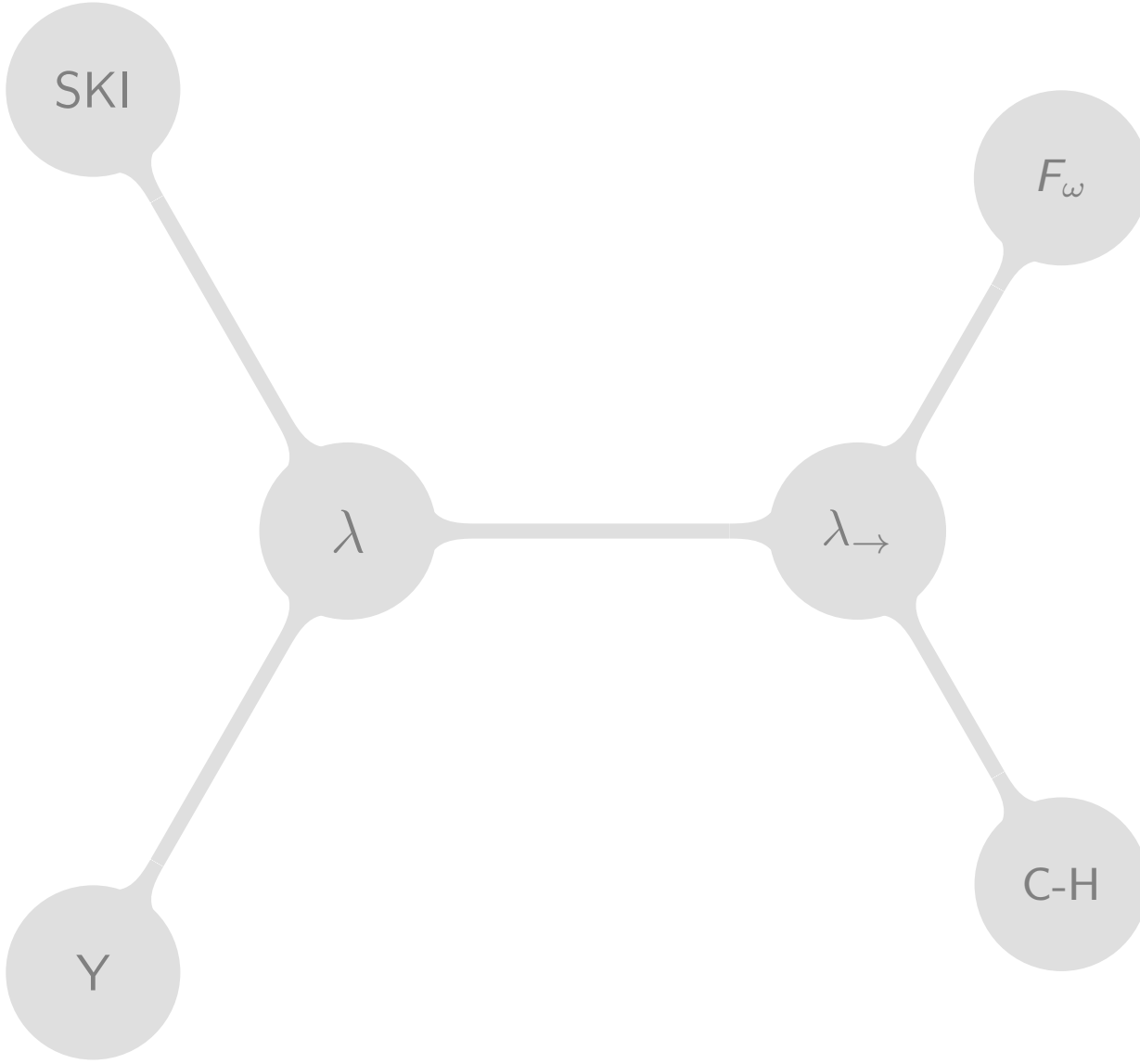
Introduction to Lambda Calculus

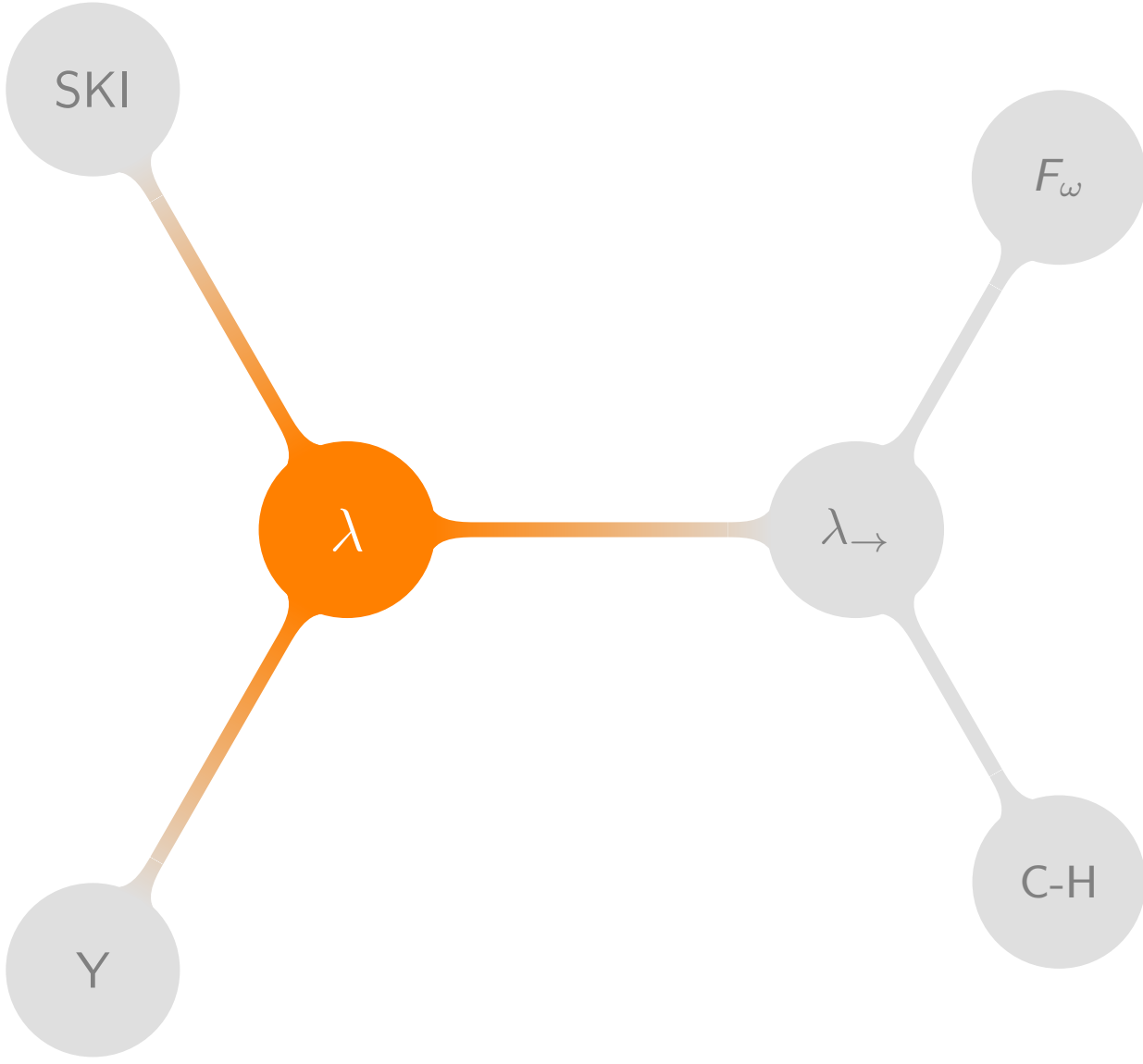
Maciek Makowski (@mmakowski)

17th October 2014

The Plan



Basic Lambda Calculus



Syntax

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

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

Syntax

v_1

Syntax

v_1

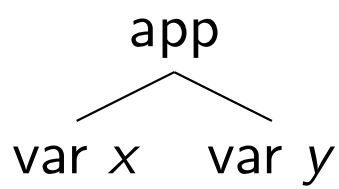
var v_1

Syntax

$x\ y$

Syntax

$x\ y$

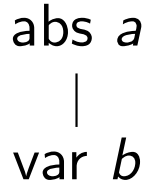


Syntax

$\lambda a.b$

Syntax

$\lambda a.b$



Syntax

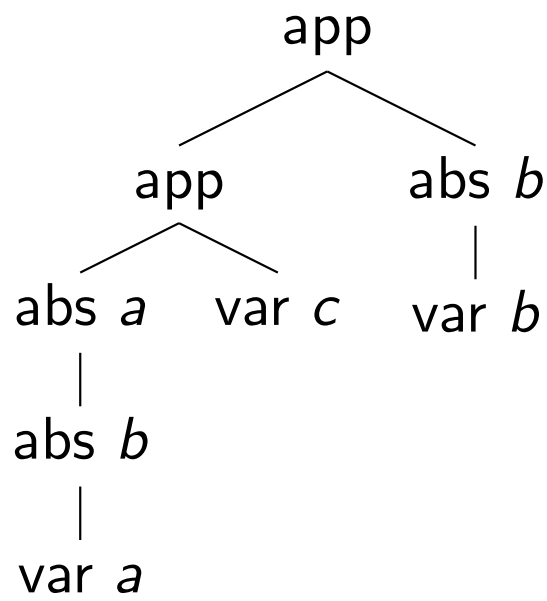
$$(\lambda a.\lambda b.a)\ c\ (\lambda b.b)$$

Syntax

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

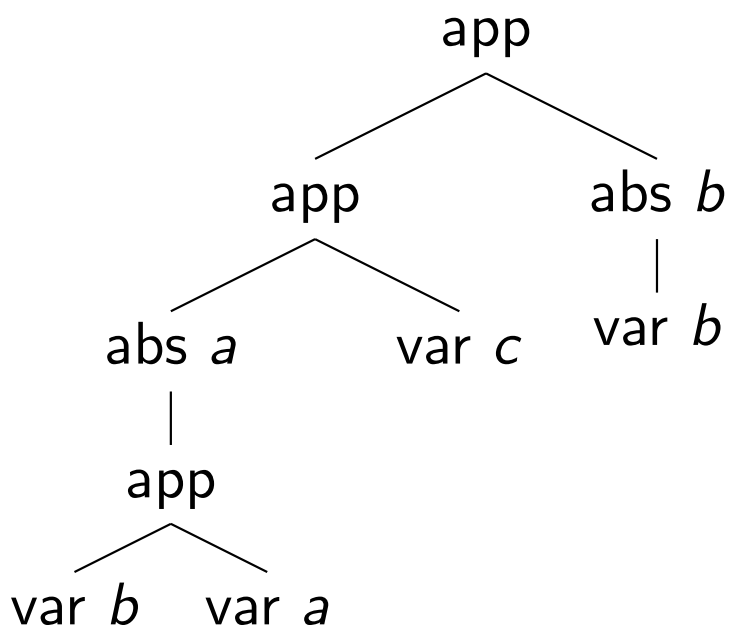
Syntax

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



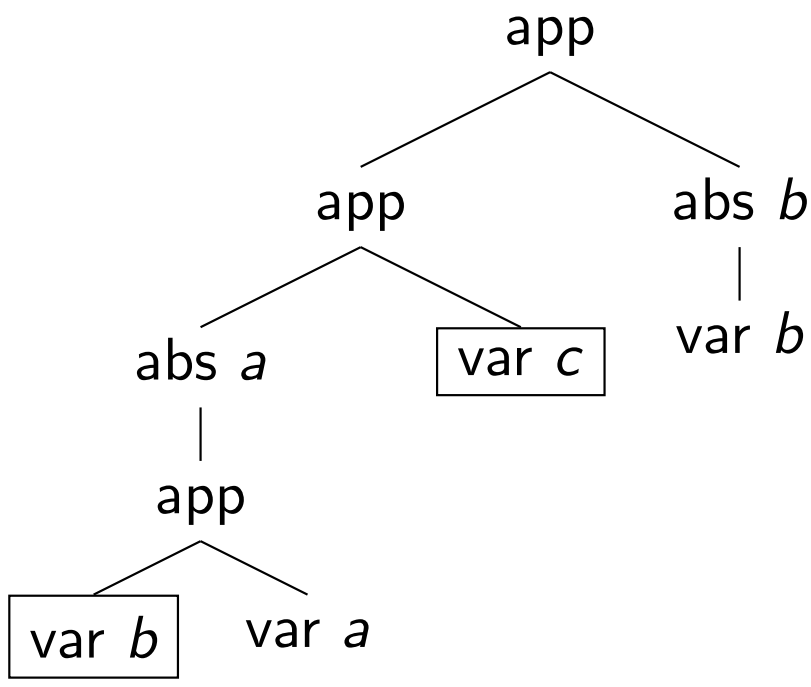
Syntax

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



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

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

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

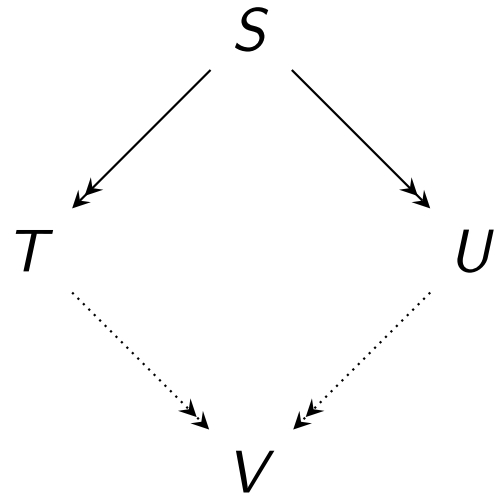
Rewriting

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



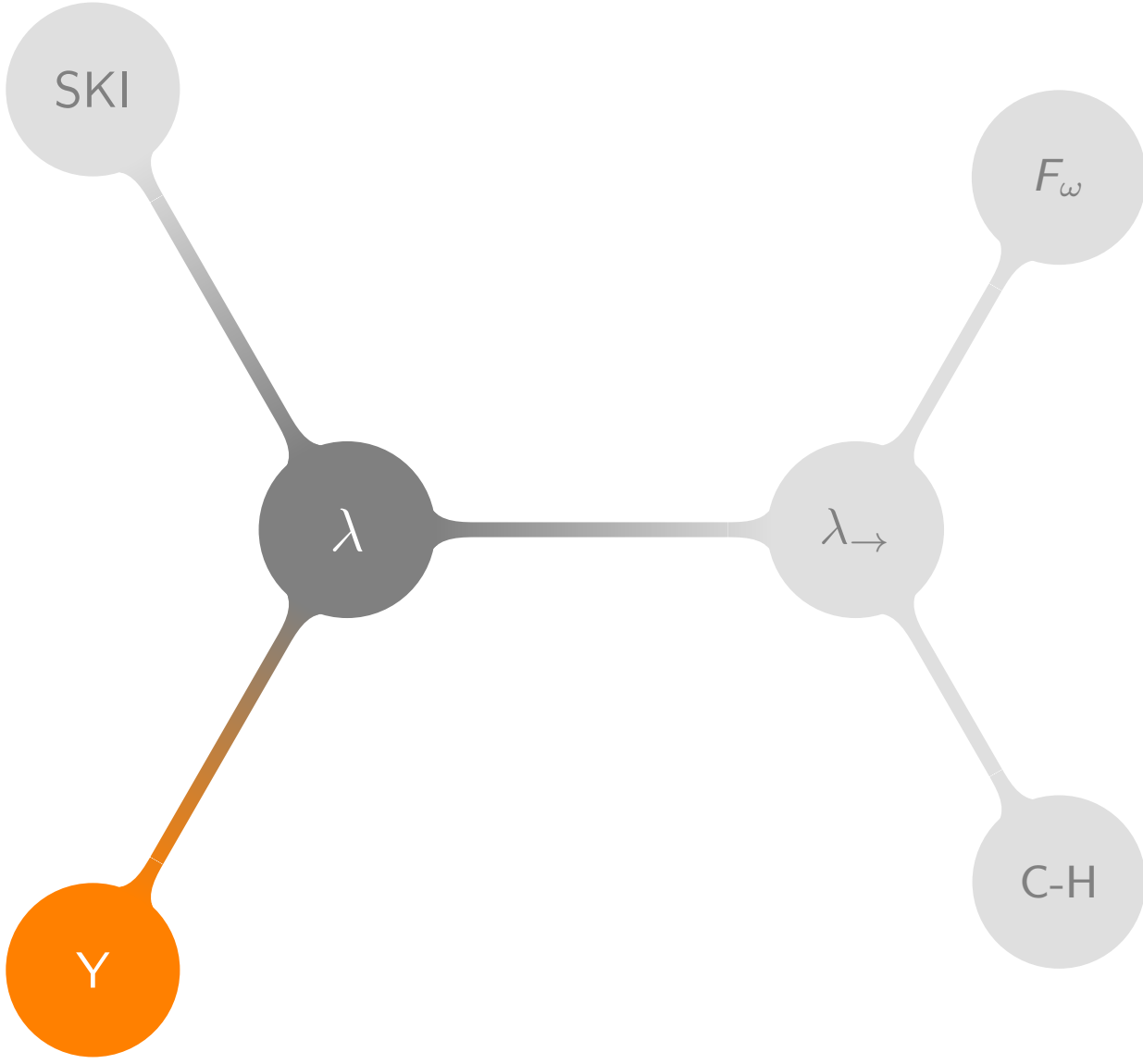
Semantics

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

Semantics

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

Programming in Lambda Calculus



Conditionals

$$\texttt{if } C \texttt{ then } T \texttt{ else } F$$

Conditionals

$$\texttt{if } C \texttt{ then } T \texttt{ else } F$$

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

Conditionals

$$\texttt{if } C \texttt{ then } T \texttt{ else } F$$

$$\begin{aligned}
\texttt{true} &= \lambda t.\lambda f.t \\
\texttt{false} &= \lambda t.\lambda f.f \\
\texttt{test} &= \lambda c.\lambda t.\lambda f.c\ t\ f \\
\texttt{if } C \texttt{ then } T \texttt{ else } F &= \texttt{test}\ C\ T\ F
\end{aligned}$$

Conditionals

$$\texttt{if } C \texttt{ then } T \texttt{ else } F$$

$$\begin{aligned}
\texttt{true} &= \lambda t.\lambda f.t \\
\texttt{false} &= \lambda t.\lambda f.f \\
\texttt{test} &= \lambda c.\lambda t.\lambda f.c\ t\ f \\
\texttt{if } C \texttt{ then } T \texttt{ else } F &= \texttt{test}\ C\ T\ F
\end{aligned}$$

Numbers

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

Numbers

$$0 = \lambda s.\lambda z.z$$

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

$$\begin{aligned}
0 &= && \lambda s.\lambda z.z \\
1 &= \mathtt{succ}\ 0 = && \lambda s.\lambda z.s\,z \\
2 &= \mathtt{succ}\ 1 = && \lambda s.\lambda z.s\,(s\,z) \\
3 &= \mathtt{succ}\ 2 = && \lambda s.\lambda z.s\,(s\,(s\,z)) \\
&\vdots \\
\mathtt{n} &= && \lambda s.\lambda z\,\underbrace{s\,(\ldots s\,(s}_{n}\ z)\ldots)
\end{aligned}$$

Numbers

$$0 = \lambda s.\lambda z.z$$

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

$$\mathtt{plus} = \lambda m.\lambda n.\lambda s.\lambda z.m\ s\ (n\ s\ z)$$

$$\mathtt{times} = \lambda m.\lambda n.m\ (\mathtt{plus}\ n)\ 0$$

Recursion

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

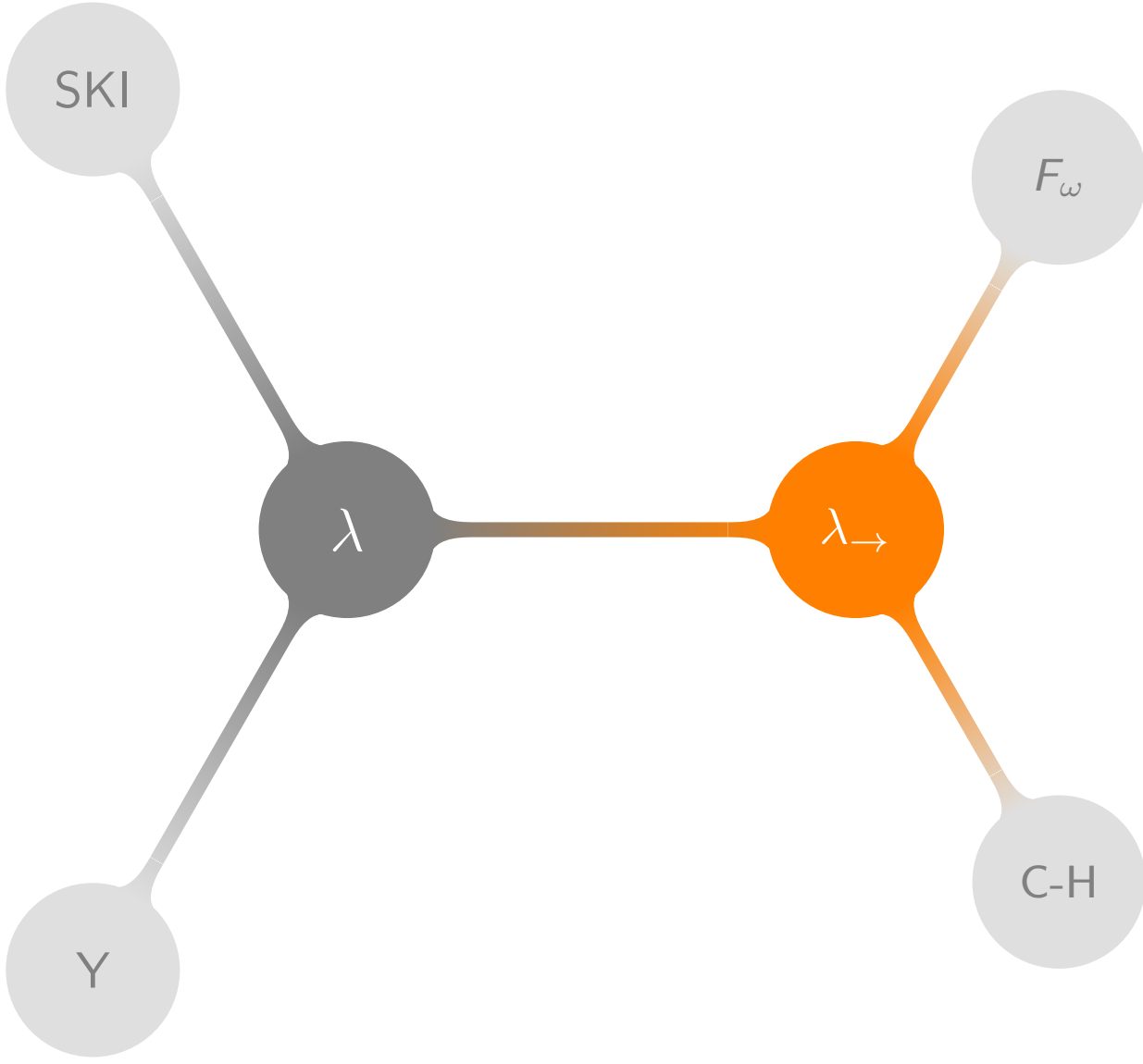
Recursion

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

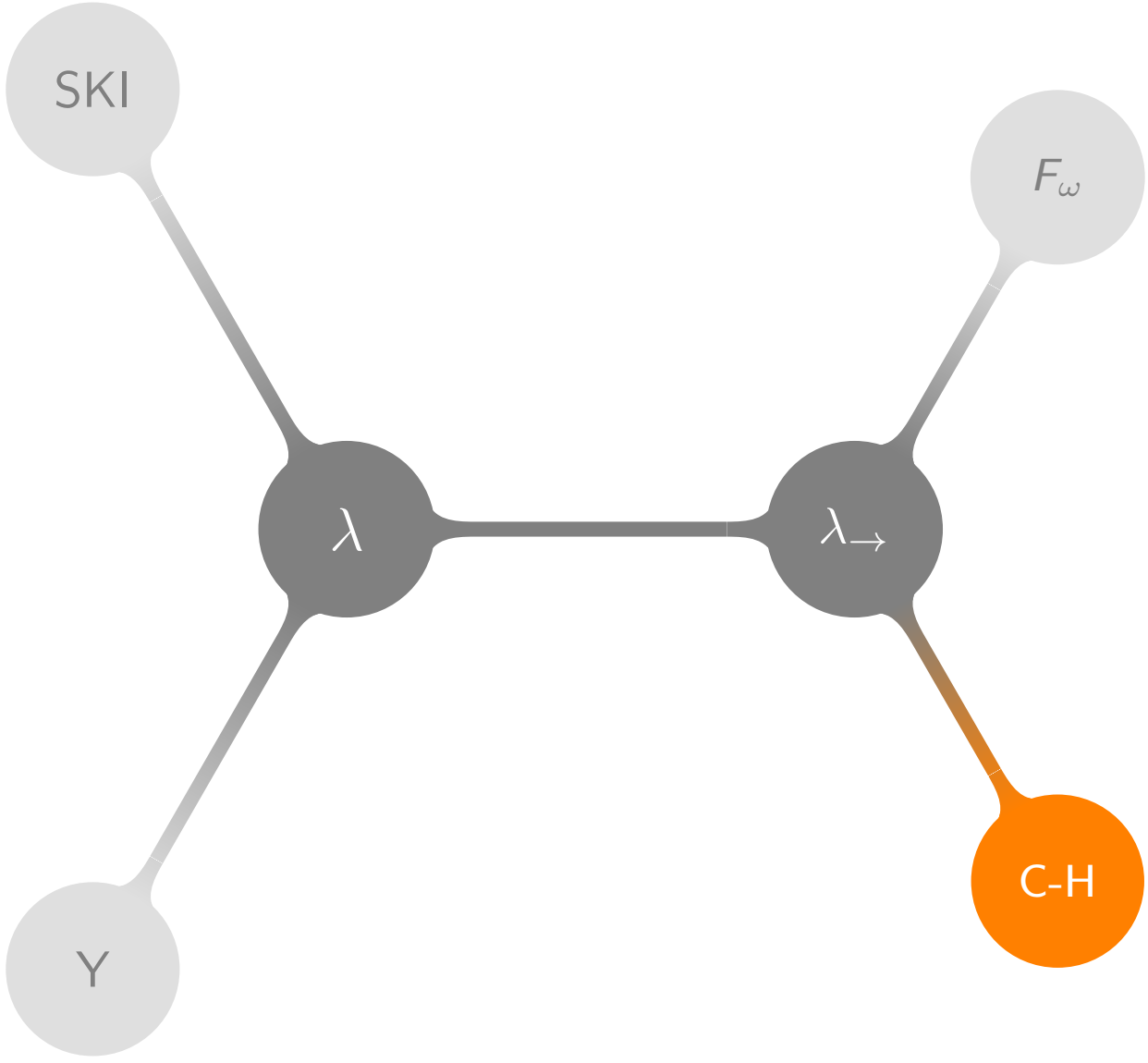
$$\mathtt{g} = \lambda f.\lambda n.\mathtt{if\ eq}\ n\ \mathtt{0\ then\ 1\ else}\ (\mathtt{times\ n}\,(f\,(\mathtt{pred\ n})))$$

$$\mathtt{factorial} = \mathtt{Y\ g}$$

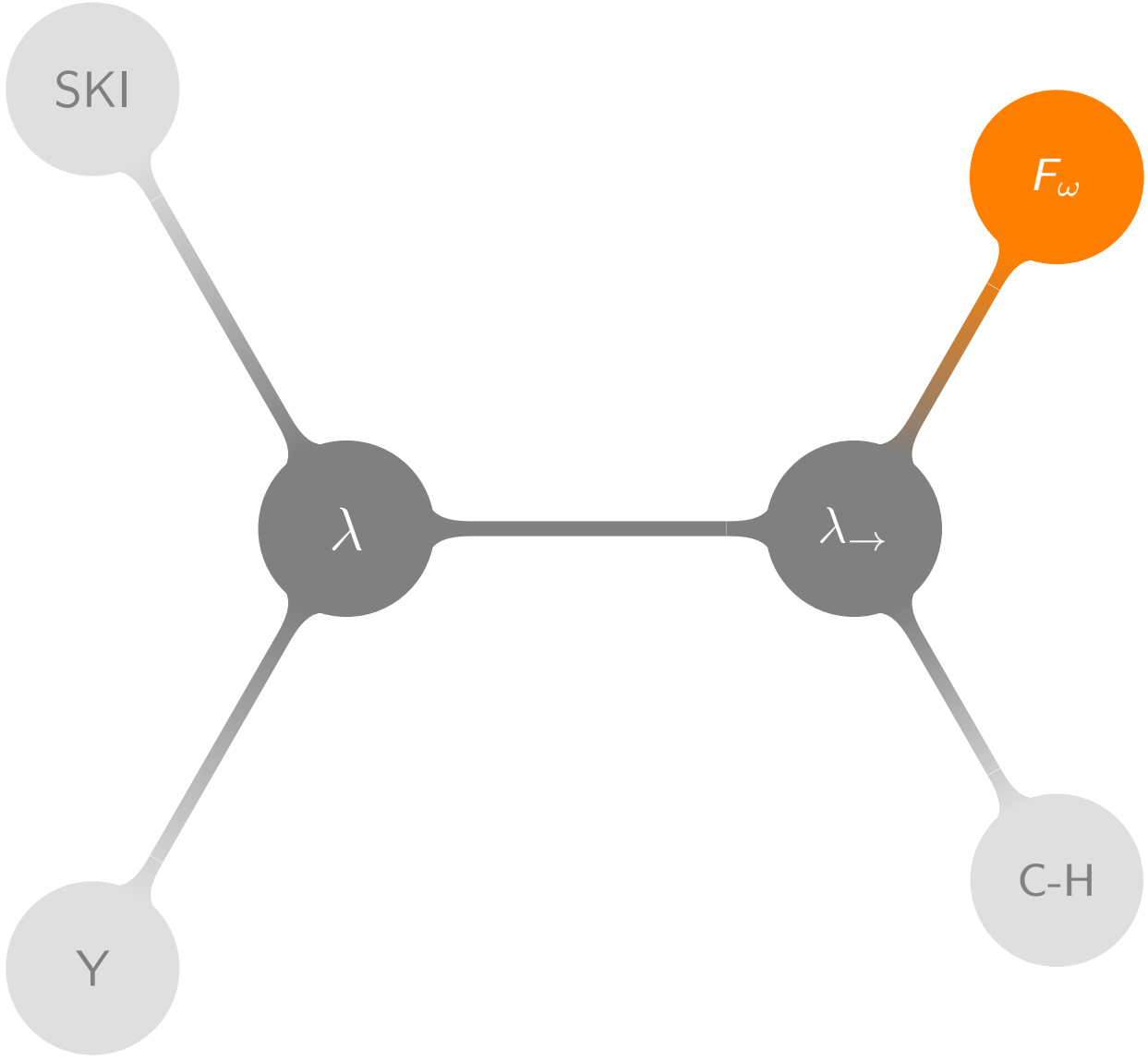
Simple Types



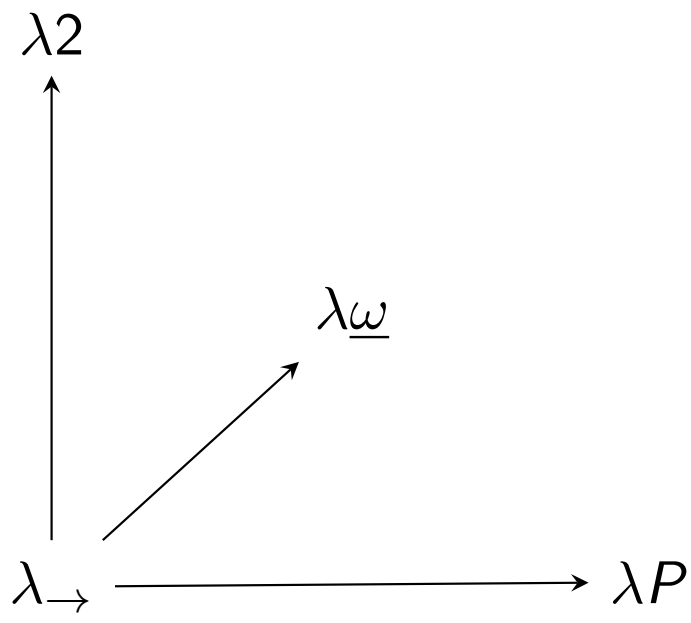
Curry-Howard Correspondence



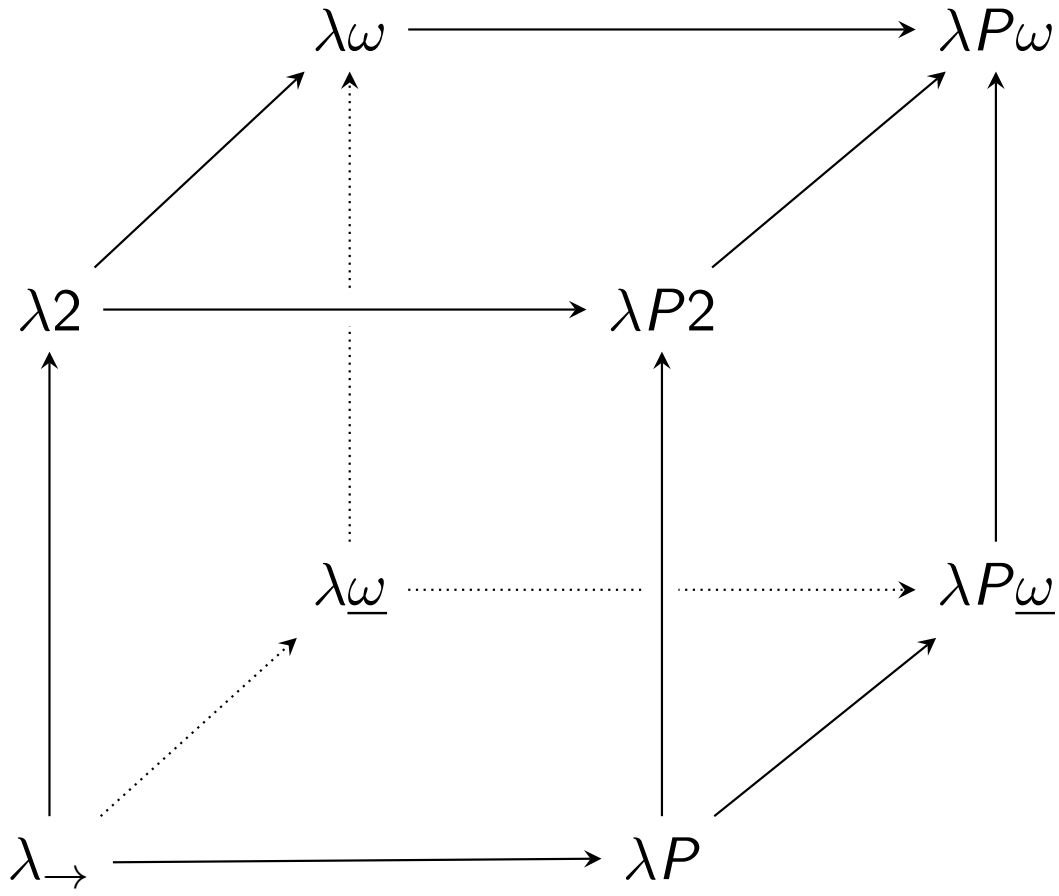
More Types



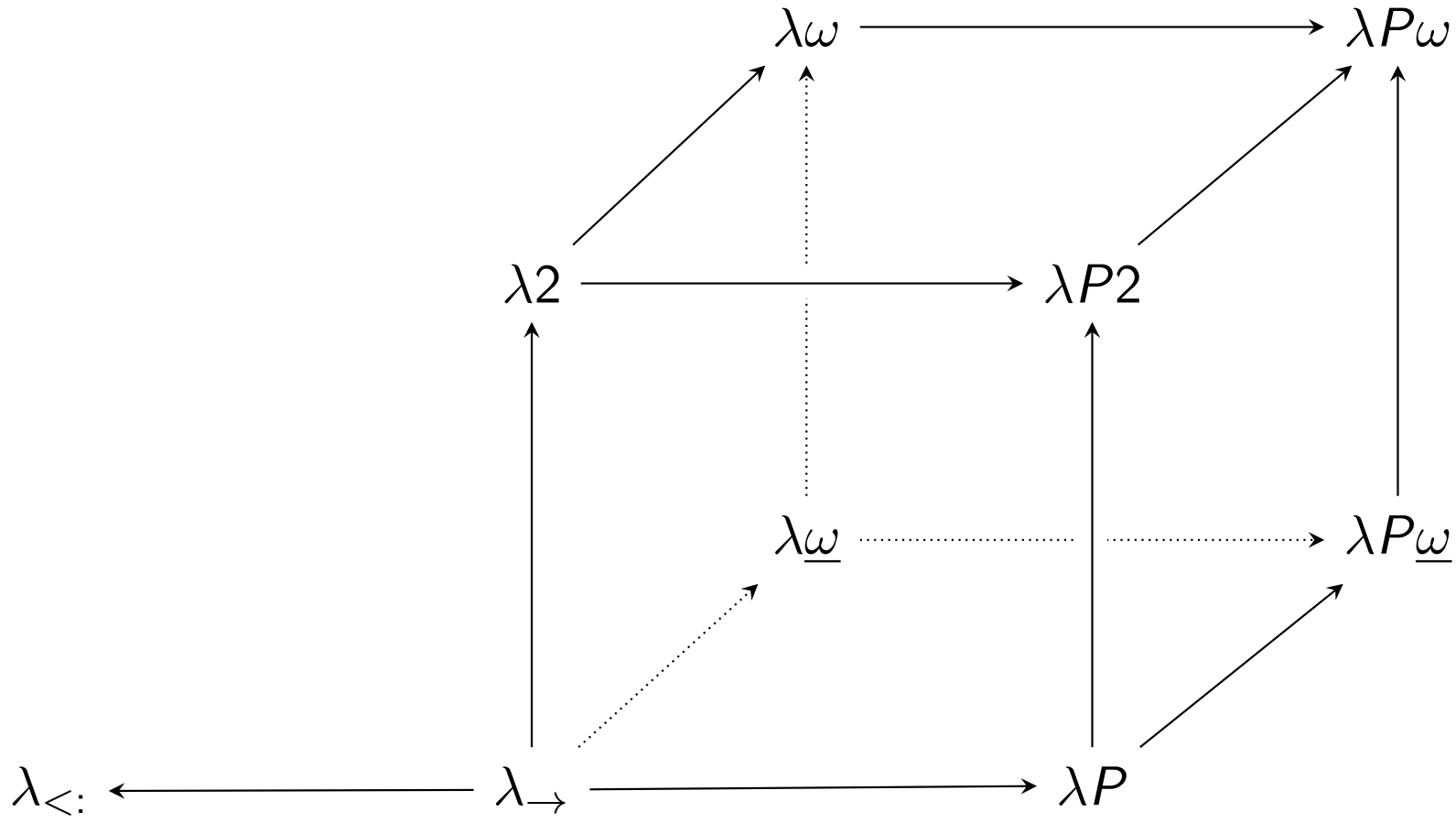
The Lambda Cube



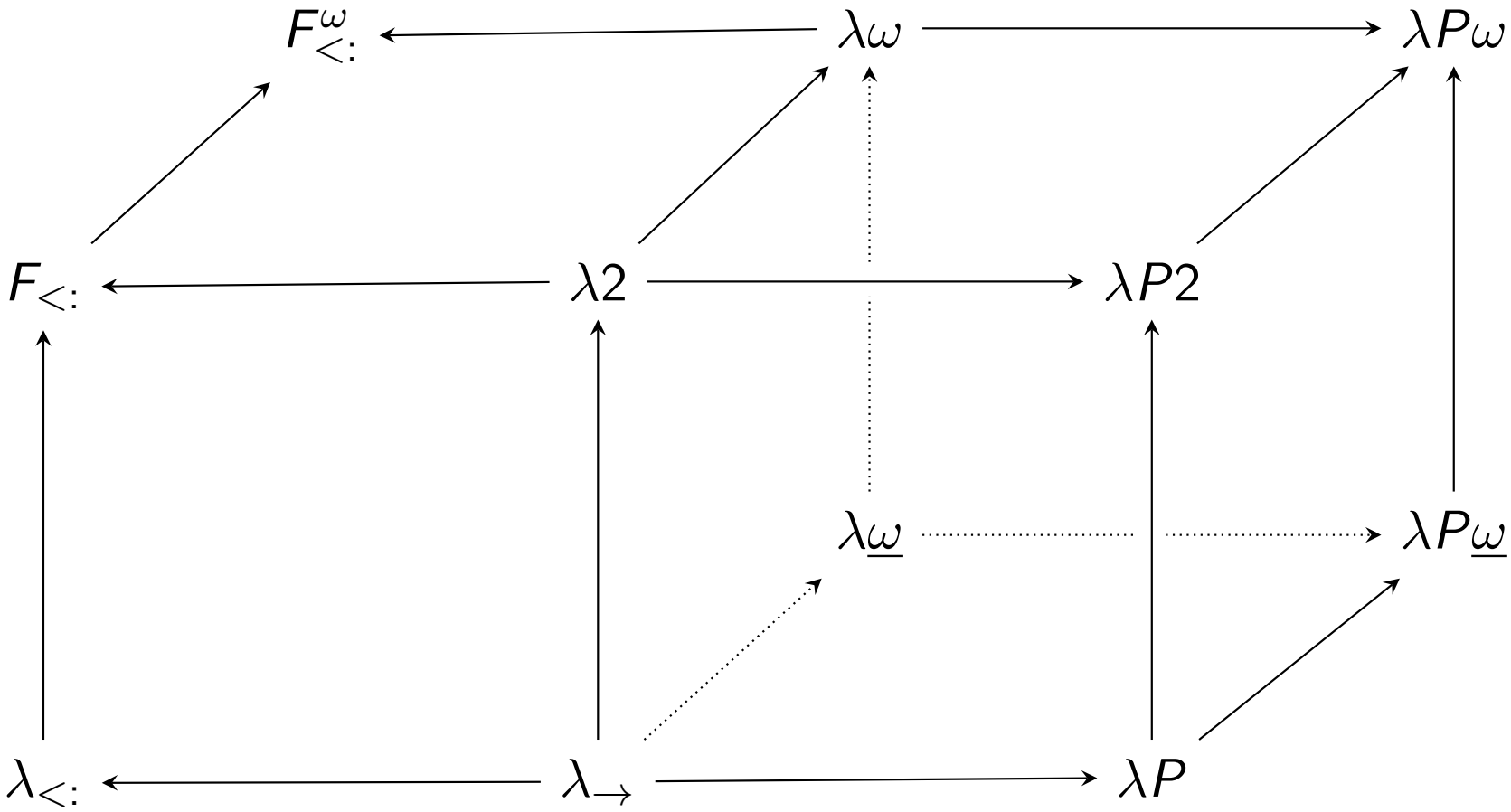
The Lambda Cube



Subtyping


$\lambda\omega$
$\lambda P\omega$
$\lambda 2$
$\lambda P2$
$\lambda\underline{\omega}$
$\lambda P\underline{\omega}$
$\lambda_{<:}$
$\lambda_{\rightarrow}$
λP

Subtyping


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