λ-calculus.

History: Frege, Schönfinkel → Church, Curry.
Foundation for mathematics? → Foundation for computation.

Syntax

$$
\begin{array}{lll}
t ::= & x & \text{— variable (any variable name)} \\
| & \lambda x.\, t & \text{— lambda aka "abstraction" aka "function".} \\
| & t_1\ t_2 & \text{— function application.}
\end{array}
$$

Ex.

x

f

$\lambda x.\, y$

$\lambda x.\, x\ y$

$(f\ (\lambda x.\ (v\ v)))\ (\lambda z.\ (\lambda w.\ z))$

Conventions:

- The body of a λ extends as far to the right as possible.
  e.g. $\lambda x.\, y\ z$ means $\lambda x.\, (y\ z)$ NOT $(\lambda x.\, y)\ z$
- Currying!
  - Multiple consecutive λ's can be abbreviated
    $\lambda x.\, \lambda y.\, \lambda z.\, x \equiv \lambda x\ y\ z.\, x$
  - Application associates to the left, so
    $f\ g\ x\ y\ z \equiv (((f\ g)\ x)\ y)\ z$

Hence,

$(f\ (\lambda x.\ (v\ v)))\ (\lambda z.\ (\lambda w.\ z)) \equiv f\ (\lambda x.\ v\ v)\ (\lambda z\ w.\ z)$

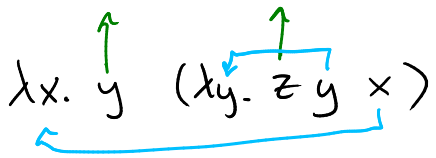
We will define rewrite rules via substitution — but we have to be careful!

① α-equivalence: the variable name in a λ doesn't matter.

e.g. $(\lambda x.\, x) \equiv (\lambda z.\, z)$

② Bound vs. free variables.

- A "bound" variable refers to an enclosing λ.
- A "free" variable does not (refers to some "external" thing).



We should only substitute for free occurrences of a variable.

We can rename via α-equivalence as necessary to avoid conflict.

Rewrite rules

$$(\lambda x.\, t_1)\; t_2 \longrightarrow [x \mapsto t_2]\, t_1$$ *

* — subst. for free x, careful w/ names.

eg.

$$(\lambda x.\, x\; y)\; (\lambda z.\, z)$$

$$\longrightarrow [x \mapsto \lambda z.\, z](x\; y) = (\lambda z.\, z)\; y$$

$$\longrightarrow [z \mapsto y]\, z = y.$$

+ Congruence rules:

- if $t_1 \longrightarrow t_1'$ then $t_1\; t_2 \longrightarrow t_1'\; t_2$
- if $t_2 \longrightarrow t_2'$ then $t_1\; t_2 \longrightarrow t_1\; t_2'$.