let $f(x) = \big(\text{let } g(x) = (x+y) \times (x+y)$ in

integrate $(0, x, g)\big)$ in

integrate $(0, 1, f)$.

According to some rumor that I heard, one motivation for introducing lambda abstraction to Java is to help people develop and use libraries with higher-order functions, in particular, collection libraries.

~~Through~~

using the lambda calculus, we will study consequences of having ~~the~~ lambda abstraction.

2. Syntax.

$$\langle exp \rangle ::= \langle var \rangle \;\big|\; \underbrace{\langle exp \rangle \langle exp \rangle}_{\text{called application}} \;\big|\; \underbrace{\lambda \langle var \rangle . \langle exp \rangle}_{\text{called abstraction or lambda expression}}$$

① examples.

$(\lambda x. x)(\lambda z. z)$

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

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

$\lambda x. \lambda y. x$ ..... encoding of true in the lambda calculus.

$\lambda x. \lambda y. y$ .... encoding of false.

$\lambda f. \lambda x. x$ .... encoding of 0

$\lambda f. \lambda x. f\, x$ .... encoding of 1.

$\lambda f. \lambda x. f\, (f\, x)$ ... encoding of 2

$\lambda f. \lambda x. f\, (f\, (f\, x))$ -- encoding of 3.

② The set of free variables, the substition operator, and (in an expression of ~~the~~ the lambda calculus) the α-equivalence (or ~~name~~ renaming equivalence) for the lambda-calculus expressions are defined as you expect. once you get the idea that ~~x~~ the lambda operator in $\lambda x.e$ binds ~~a~~ the variable x in the expression e, just as the quantifier $\forall$ in $\forall x. p$ binds the variable x in an assertion p.

i)

$FV(v) = \{v\}$

$FV(e_1\, e_2) = FV(e_1) \cup FV(e_2)$

$FV(\lambda v. e) = FV(e) \setminus \{v\}$.

ii)

δ is a substitution, i.e. a map from $\langle var \rangle$ to $\langle exp \rangle$.

$v / \delta \;=\; \delta(v)$

$(e_1\, e_2) / \delta \;=\; (e_1 / \delta)\, (e_2 / \delta)$

$(\lambda v. e) / \delta \;=\; \lambda v_{new}. \big(e / [\delta \mid v : v_{new}] \big)$.

This means the substitution that maps all variables to themselves except v, which it maps to v_{new}

where $v_{new} \notin \bigcup_{w \in FV(e) \setminus \{v\}} FV(\delta(w))$.

iii) The renaming or change of bound variable means ~~an~~ the operation of ~~re~~ replacing ~~a bound~~ an occurrence of $\lambda v. e$ by $\lambda v_{new}. (e / v \to v_{new})$ for $v_{new} \notin FV(e) \setminus \{v\}$. Two expressions e_1 and e_2 are α-equivalent or renaming-equivalent if we can obtain e_2 from e_1 by applying this renaming operation to some subexpressions of e_1 zero or multiple times. We write $e_1 \equiv e_2$ to denote their α-equivalence.

Example. $(\lambda x. x)(\lambda z. z) \;\equiv\; (\lambda x. x)(\lambda y. y)$

$(\lambda x. \lambda y. x) \;\equiv\; (\lambda y. \lambda x. y)$