λ Calculus Worksheet

Angel Alvarez
Foundations of Computer Science, λ Calculus

November 15, 2017

Definition 1.0.0. The Definition of λ Calculus

- Let Λ be the set of all valid expressions in λ Calculus.
- Let $x \in A$ mean that x is a member of the symbol-space (alphabet) A.
- Let $e \in \Lambda$ mean that e is a valid expression in the set of all valid λ Calculus expressions.

$$\forall x \in A.\, x \in \Lambda \qquad \forall x \in A, \forall M \in \Lambda.\, \lambda x.M \in \Lambda \qquad \forall M, N \in \Lambda.\, (M\, N) \in \Lambda$$

We can also define λ Calculus expressions like so:

$$e ::= x \mid (\lambda x.e) \mid (e\, e)$$

Exercise 1.0.1. Determine whether the following are valid λ Calculus expressions:

$$\lambda x.x \qquad \lambda a.b \qquad x.x \qquad \lambda\lambda.\lambda \qquad \lambda x.h\,e\,l\,l\,o\,w\,o\,r\,l\,d$$

$$\lambda x.\lambda y.x + y \qquad \lambda x \qquad \lambda\lambda.x \qquad \lambda.. \qquad \lambda h.\lambda e.\lambda l.\lambda o.\lambda w.\lambda r.\lambda d.h\,e\,l\,l\,o\,w\,o\,r\,l\,d$$

Definition 1.1.0. The Definition of a Combinator

- Let $x \in A$. x is a closed variable *IFF* is attached to a λ expression.
- Let $y \in A$. y is a free variable *IFF* is not a closed variable.
- Let $e \in \Lambda$. e is a combinator *IFF* all variables in e are closed variables.

Exercise 1.1.1. Determine whether the following are valid combinators:

$$\lambda x.x \qquad \lambda y \qquad \lambda x.\lambda y.y \qquad \lambda x.h\,e\,l\,l\,o\,w\,o\,r\,l\,d$$

$$\lambda x.y \qquad \lambda a.b \qquad \lambda b.\lambda a.a\,b \qquad \lambda h.\lambda e.\lambda l.\lambda o.\lambda w.\lambda r.\lambda d.h\,e\,l\,l\,o\,w\,o\,r\,l\,d$$

Lemma 2.0.0. α Equivalence

- Let $\lambda x.M[y]$ represent an expression where all instances of x in M are replaced with the expression y.

$$(\lambda x.M[x]) \Rightarrow (\lambda y.M[y])$$

Do not to replace variables with symbols already in use in your expression.

Exercise 2.0.1. Determine whether the following are valid uses of α equivalence:

$$\lambda x.\lambda y.x \Rightarrow \lambda y.\lambda y.y \quad \lambda x.\lambda y.y \Rightarrow \lambda a.\lambda y.y \quad \lambda i.\lambda j.i\,j\,j \Rightarrow \lambda k.\lambda j.j\,k\,k$$

$$\lambda x.x \Rightarrow \lambda f.f \qquad \lambda f.f \Rightarrow \lambda f f.(f\,f) \qquad \lambda a.\lambda b.a \Rightarrow \lambda x.\lambda y.x$$

Lemma 2.1.0. β Reduction

- Let $\lambda x.M[y]$ represent an expression where all instances of x in M are replaced with the expression y.

$$(\lambda x.M[x])\,E \Rightarrow (M[x ::= E])$$

Exercise 2.1.1. apply the β reduction rules to the following expressions. **Show your work!**

1. $(\lambda w.w)\,(\lambda x.x)\,(\lambda y.y)\,(\lambda z.z)$

2. $(\lambda x.\lambda y.y)\,(\lambda a.\lambda b.b)\,(\lambda f.\lambda g.f)$

3. $(\lambda f.\lambda x.f\,x)\,(\lambda x.x)\,(\lambda f.\lambda x.f\,(f\,(f\,x)))$

Church Encodings of truth values:

- Let $True ::= \lambda x.\lambda y.x$
- Let $False ::= \lambda x.\lambda y.y$

Logic operators on the encodings:

- Let $Not ::= \lambda x.x\, False\, True$
- Let $And ::= \lambda x.\lambda y.x\;\; y\, False$
- Let $Or ::= \lambda x.\lambda y.x\, True\, y$
- Let $\mathit{IfThenElse} ::= \lambda x.\lambda y.\lambda z.x\, y\, z$

Exercise 3.0.0. Simplify the following logical statements, first by expressing the leftmost expression in λ Calculus, and then reducing the expressions.

1. $(Not\ \ True\ \ False\ \ True)$

2. $(\mathit{And}\ \ (Or\ \ False\ \ True)\ \ True)$

3. $(\mathit{IfThenElse}\ \ True\ \ False\ \ True)$

Church Encodings of Integers

- Let $Zero ::= \lambda f.\lambda x.x$
- Let $One ::= \lambda f.\lambda x.f\ x$
- Let $Two ::= \lambda f.\lambda x.f\ (f\ x)$
- Let $Three ::= \lambda f.\lambda x.f\ (f\ (fx))$
- Let $Four ::= \lambda f.\lambda x.f\ (f\ (f\ (f\ x)))$
- Let $Five ::= \lambda f.\lambda x.f\ (f\ (f\ (f\ (f\ x))))$
- ...

Mathematical Operators:

- Let $Add ::= \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x)$
- Let $Mult ::= \lambda m.\lambda n.\lambda f.m\ (n\ f)$
- Let $Exp ::= \lambda m.\lambda n.n\ m$
- Let $Succ ::= \lambda n.\lambda f.\lambda x.f\ (n\ f\ x)$
- Let $Pred ::= \lambda n.\lambda f.\lambda x.n\,(\lambda g.\lambda h.h\,(g\,f))\ (\lambda u.x)\ (\lambda u.u)$
- Let $Sub ::= \lambda m.\lambda n.n\ Pred\, m$

Exercise 4.0.0. Simplify the following mathematical statements by reducing them in λ Calculus.

1. $(Mult\ \ Zero\ \ Two)$

2. $(Add\ \ Three\ \ (Succ\ \ One))$

Definition 5.0.0. Definition of the Y Combinator

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

Exercise 5.0.1. Let $FOO ::= \lambda f.\lambda n.\lambda m.isZero\,n\,m\,(f\,(Pred\,n)\,(Succ\,m))$
Simplify the following recursive function call. Don't worry about decoding every expression. If something is obvious, such as $(Pred\,Three)$, then you don't need to show that step fully. Remember that isZero $::= \lambda n.n\,(\lambda x.False)\,True$

Hint: You can give some expression a name to simplify your work. I recommend using the following simplification: RecF $::= (\lambda x.FOO\,(x\,x))(\lambda x.FOO\,(x\,x))$. Knowing what this does will make this evaluation significantly easier.

- $((Y\,FOO)\,Two\,Three)$

Does this function remind you of any other function we have studied?