Notes on Lambda Calculus

Syntax

$e ::= x$		Variable
$\mid$	$\lambda x.\, e$	Abstraction
$\mid$	$e_1\ e_2$	Application

Examples:

x	A variable x
$\lambda x.\, \lambda y.\, x$	A function that returns a function that returns the argument of the outer function.
$(\lambda x.\, \lambda y.\, x)\ a\ b$	Apply a to the outer function, then b to the inner function.

Convention: $\lambda x.\, \lambda y.\, x$ is interpreted as $\lambda x. (\lambda y.\, x)$

$f\ a\ b$ " $(f\ a)\ b$

Reducing Lambda Calculus Expressions

There is only one "operation" in lambda calculus: apply a function to an argument.
We do this by substituting the argument into the function body.

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

$$(\lambda x.\ \lambda y.\ x)\ a \quad \to \quad (\lambda y.\ a)\ b \to a$$

There are two subtleties when substituting, mainly due to "shadowing" variables.
First is using a shadowed variable inside an abstraction:

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

Variables refer to the "closest" binding.

$$\to (\lambda x. x)\ y$$

We can determine if we're shadowing by checking the set of "free variables", i.e. unbound variables

$$FV(x) = \{x\}$$ x by itself is not bound.

$$FV(\lambda x.\ e) = FV(e) - \{x\}$$ x is bound here

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

We should only substitute into a lambda if the lambda's variable is different from the variable we're substituting, i.e. if $x \neq y$ when substituting x in $(\lambda y.\ e)$

Second subtlety is when applying a shadowed variable:

$$(\lambda x. \lambda y. x\ y)\ y \to ?$$

We can't just substitute y directly because it will change what the inner function does.

$(\lambda y. y\ y)$ is wrong! (Can you think of an example that shows this?)

Instead, note that we can rename the variable of the inner abstraction without changing the semantics of the inner function. This gives the correct answer:

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

As a last note, there could be multiple applications that you can perform at a time.
The strategy for picking which one to reduce first is called a reduction strategy.
This will be covered in Monday's lecture.

Exercises

Evaluate the following until you cannot make any more reductions (or try to).

1. $(\lambda x. \lambda y. y)\ (\lambda x. x)\ (\lambda y. y\ y)$
2. $(\lambda x. \lambda y. x)\ (\lambda x. x)\ (\lambda y. y\ y)$
3. $(\lambda x. x\ x)\ (\lambda x. x\ x)$

   Is there anything you notice about this?

4. $(\lambda x. f(x\ x))\ (\lambda x. f(x\ x))$

   Try reducing inside the outer lambda. What's interesting about this version?

5. $(\lambda n. \lambda f. \lambda x. f\ (n\ f\ x))\ (\lambda f. \lambda x. x)\ (\lambda x. x * 2)\ 1$

Solution to Exercises

Note: this is by reducing outer expressions first.

1. $\underbrace{(\lambda x. \lambda y. y)\ (\lambda x. x)}\ (\lambda y. y\ y)$

Step 1: Reduce the left-most application.
Substitute x in $(\lambda y. y)$ with $(\lambda x. x)$

$\rightarrow (\lambda y. y)\ (\lambda y. y\ y)$

Step 2: Reduce again.
Substitute y in y with $(\lambda y. y\ y)$

$\rightarrow \lambda y. y\ y$

2. $\underbrace{(\lambda x. \lambda y. x)\ (\lambda x. x)}\ (\lambda y. y\ y)$

Step 1: Reduce the left-most application
Substitute x in $(\lambda y. x)$ with $(\lambda x. x)$

$\rightarrow (\lambda y. (\lambda x\ x))\ (\lambda y. y\ y)$

Step 2: Reduce the application.
Substitute y in $(\lambda x. x)$ with $(\lambda y. y y)$

$\rightarrow \lambda x. x$

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

Step 1: Reduce the application.

Substitute x in $(x\ x)$ with $(\lambda x.\ x x)$

$\rightarrow (\lambda x.\ x\ x)\ (\lambda x.\ x\ x)$

Note that we got the same thing back.
It's not possible to reduce this fully.
We call this expression Ω, the diverging operator.

4. $(\lambda x.\ f(x\ x))\ (\lambda x.\ f(x\ x))$

Step 1: Reduce the application

Substitute x in $f\ (x\ x)$ with $(\lambda x. f\ (x x))$

$\rightarrow f\left((\lambda x. f\ (x\ x))\ (\lambda x. f\ (x\ x))\right)$

Note that the argument to the outer f is what we started with!

If we try to reduce further, we end up with a chain of

$f(f(f(\cdots)))$

This is called the fixed-point combinator or the Y combinator.

The Y-combinator can be used to write recursive functions.

5. $\underbrace{(\lambda n. \lambda f. \lambda x. f\ (n\ f\ x))\ (\lambda f. \lambda x.\ x)}\ (\lambda x.\ \underline{x * 2})\ 1$

Assume "*" is merely a symbol with no semantics

Step 1: Reduce the leftmost application.

Substitute n in $(\lambda f. \underbrace{\lambda x. f (n\ f\ x)})$ with $(\lambda f. \lambda x.\ x)$

$\to (\lambda f. \lambda x.\ f\ ((\lambda f. \lambda x.\ x)\ f\ x)))\ (\lambda x.\ x * 2)\ 1$

Make sure the variables refer to the correct bindings!

Original: $(\lambda n. \lambda f. \lambda x.\ f\ (n\ f\ x))$

Substituted: $(\lambda f. \lambda x.\ f\ ((\lambda f. \lambda x.\ x)\ f\ x)))$

Step 2: Reduce the leftmost application.

Substitute f in $(\lambda x. f\ ((\lambda f. \lambda x.\ x)\ f\ x))$ with $(\lambda x.\ x * 2)$

Be wary of shadowing here.

$$\to (\lambda x.\ (\lambda x.\ x * 2)\ ((\lambda f.\ \underline{\lambda x.\ x})\ (\lambda x.\ x * 2)\ x))\ 1$$

Do not substitute f in here.

Step 3: Reduce the application
Substitute x with 1.
Be wary of shadowing.

$$\to (\lambda x.\ x * 2)\ ((\lambda f.\ \lambda x.\ x)\ (\lambda x.\ x * 2)\ 1))$$

These x's are all bound, so do not substitue here.

Step 4: Reduce

$$\to ((\lambda f.\ \lambda x.\ x)\ (\lambda x.\ x * 2)\ 1) * 2$$

Step 5: Reduce

$$\to ((\lambda x.\ x)\ 1) * 2$$

$$\to 1 * 2$$