CS 4110

Programming Languages & Logics

Lecture 14
More λ-calculus

29 September 2014

Announcements

- PS #4 due Wednesday
- Foster office hours 4-5pm
- Wednesday: CS 50 and Gates Dedication! No lecture
- Next Monday: Preliminary Exam I

Review: λ-calculus

Syntax

$$
\begin{array}{lcl}
e & ::= & x \mid e_1\; e_2 \mid \lambda x.\, e \\
v & ::= & \lambda x.\, e
\end{array}
$$

Semantics

$$
\frac{e_1 \rightarrow e_1'}{e_1\; e_2 \rightarrow e_1'\; e_2} \qquad \frac{e \rightarrow e'}{v\; e \rightarrow v\; e'}
$$

$$
\frac{}{(\lambda x.\, e)\; v \rightarrow e\{v/x\}}\; \beta
$$

Example: Twice

Consider the function defined by $\textit{double}\ x = x + x$.

Example: Twice

Consider the function defined by $\textit{double}\ x = x + x$.

Now suppose we want to apply *double* multiple times:

Example: Twice

Consider the function defined by $\mathit{double}\ x = x + x$.

Now suppose we want to apply *double* multiple times:

$$\mathit{quadruple}\ x \quad = \quad \mathit{double}\ (\mathit{double}\ x)$$

Example: Twice

Consider the function defined by $\mathit{double}\ x = x + x$.

Now suppose we want to apply *double* multiple times:

$$\begin{aligned} \mathit{quadruple}\ x &= \mathit{double}\ (\mathit{double}\ x) \\ \mathit{octuple}\ x &= \mathit{quadruple}\ (\mathit{quadruple}\ x) \end{aligned}$$

Example: Twice

Consider the function defined by $\mathit{double}\ x = x + x$.

Now suppose we want to apply *double* multiple times:

$$\begin{aligned}
\mathit{quadruple}\ x &= \mathit{double}\ (\mathit{double}\ x) \\
\mathit{octuple}\ x &= \mathit{quadruple}\ (\mathit{quadruple}\ x) \\
\mathit{hexadecatuple}\ x &= \mathit{octuple}\ (\mathit{octuple}\ x)
\end{aligned}$$

Example: Twice

Consider the function defined by $\mathit{double}\ x = x + x$.

Now suppose we want to apply *double* multiple times:

$$\begin{aligned} \mathit{quadruple}\ x &= \mathit{double}\ (\mathit{double}\ x) \\ \mathit{octuple}\ x &= \mathit{quadruple}\ (\mathit{quadruple}\ x) \\ \mathit{hexadecatuple}\ x &= \mathit{octuple}\ (\mathit{octuple}\ x) \end{aligned}$$

We can abstract this pattern using a generic function:

$$\mathit{twice} \triangleq \lambda f.\ \lambda x.\ f\ (f\ x)$$

Example: Twice

Consider the function defined by $\mathit{double}\, x = x + x$.

Now suppose we want to apply *double* multiple times:

$$\begin{aligned} \mathit{quadruple}\, x &= \mathit{double}\, (\mathit{double}\, x) \\ \mathit{octuple}\, x &= \mathit{quadruple}\, (\mathit{quadruple}\, x) \\ \mathit{hexadecatuple}\, x &= \mathit{octuple}\, (\mathit{octuple}\, x) \end{aligned}$$

We can abstract this pattern using a generic function:

$$\mathit{twice} \triangleq \lambda f.\ \lambda x.\ f\,(f\,x)$$

Now the functions above can be written as

$$\begin{aligned} \mathit{quadruple}\, x &= \mathit{twice}\ \mathit{double} \\ \mathit{octuple}\, x &= \mathit{twice}\ \mathit{quadruple} \\ \mathit{hexadecatuple}\, x &= \mathit{twice}\ \mathit{octuple} \\ & \quad (\text{or } \mathit{twice}\, (\lambda x.\ \mathit{twice}\, x)) \end{aligned}$$

Evaluation

The essence of λ-calculus evaluation is the β-reduction rule, which says how to apply a function to an argument.

$$\frac{}{(\lambda x.\, e)\; v \rightarrow e\{v/x\}}\ \beta\text{-reduction}$$

But there are many different evaluation strategies, each corresponding to particular ways of using β within terms:

- Full β reduction
- Call-by-value
- Call-by-name
- Normal order
- etc.

Call by value

$$\frac{e_1 \to e_1'}{e_1\ e_2 \to e_1'\ e_2} \qquad \frac{e_2 \to e_2'}{v_1\ e_2 \to v_1\ e_2'}$$

$$\frac{}{(\lambda x.\, e_1)\ v_2 \to e_1\{v_2/x\}}\ \beta$$

Key characteristics:

- Arguments evaluated fully before they are supplied to functions
- Evaluation goes from left to right (in this presentation)
- We don't evaluate "under a λ"

Call by name

$$\frac{e_1 \to e_1'}{e_1\ e_2 \to e_1'\ e_2}$$

$$\frac{}{(\lambda x.\, e_1)\ e_2 \to e_1\{e_2/x\}}\ \beta$$

Key characteristics:

- Arguments supplied immediately to functions
- Evaluation still goes from left to right (in this presentation)
- We still don't evaluate "under a λ"

Question

Fully evaluating any λ-calculus term under call by value and call by name will produce the same result?

A. True

B. False

Full β reduction

$$\frac{e_1 \rightarrow e_1'}{e_1\ e_2 \rightarrow e_1'\ e_2} \qquad \frac{e_2 \rightarrow e_2'}{e_1\ e_2 \rightarrow e_1\ e_2'}$$

$$\frac{e \rightarrow e'}{\lambda x.\ e \rightarrow \lambda x.\ e'}$$

$$\frac{}{(\lambda x.\ e_1)\ e_2 \rightarrow e_1\{e_2/x\}}\ \beta$$

Key characteristics:

- Use the β rule anywhere...
- ...including "under a λ"

Question

Which of the following strategies terminate?

A. Call by value
B. Call by name
C. Full β reduction
D. All of them
E. None of them

Question

Which of the following strategies are non-deterministic?

A. Call by value
B. Call by name
C. Full β reduction
D. All of them
E. None of them

Question

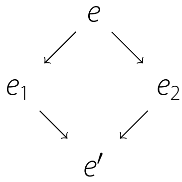
Does the non-determinism in full β reduction affect the final result?

A. Yes

B. No

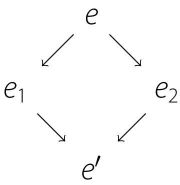
Confluence

Full β reduction has the following property:



Confluence

Full β reduction has the following property:



Theorem (Confluence)

If $e \rightarrow^* e_1$ and $e \rightarrow^* e_2$ then $e_1 \rightarrow^* e'$ and $e_2 \rightarrow^* e'$ for some e'.

Substitution

The main workhorse in the β rule is substitution, which replaces free occurrences of a variable x with a term e

However, defining substitution correctly is actually quite subtle

Substitution I

As a first attempt, consider the following:

Substitution I

As a first attempt, consider the following:

$$y\{e/x\} = \begin{cases} e & \text{if } y = x \\ y & \text{otherwise} \end{cases}$$

Substitution I

As a first attempt, consider the following:

$$
\begin{array}{rcl}
y\{e/x\} & = & \begin{cases} e & \text{if } y = x \\ y & \text{otherwise} \end{cases} \\
(e_1\ e_2)\{e/x\} & = & (e_1\{e/x\})\ (e_2\{e/x\})
\end{array}
$$

Substitution I

As a first attempt, consider the following:

$$
\begin{aligned}
y\{e/x\} &= \begin{cases} e & \text{if } y = x \\ y & \text{otherwise} \end{cases} \\
(e_1\ e_2)\{e/x\} &= (e_1\{e/x\})\ (e_2\{e/x\}) \\
(\lambda y.e_1)\{e/x\} &= \lambda y.e_1\{e/x\}
\end{aligned}
$$

Substitution I

As a first attempt, consider the following:

$$
\begin{array}{rcl}
y\{e/x\} & = & \begin{cases} e & \text{if } y = x \\ y & \text{otherwise} \end{cases} \\
(e_1\ e_2)\{e/x\} & = & (e_1\{e/x\})\ (e_2\{e/x\}) \\
(\lambda y.e_1)\{e/x\} & = & \lambda y.e_1\{e/x\}
\end{array}
$$

What's wrong with this definition?

Substitution I

As a first attempt, consider the following:

$$
\begin{array}{rcl}
y\{e/x\} & = & \begin{cases} e & \text{if } y = x \\ y & \text{otherwise} \end{cases} \\
(e_1\ e_2)\{e/x\} & = & (e_1\{e/x\})\ (e_2\{e/x\}) \\
(\lambda y.e_1)\{e/x\} & = & \lambda y.e_1\{e/x\}
\end{array}
$$

What's wrong with this definition?

It substitutes bound variables too!

$$(\lambda y.y)\{3/y\}$$

Substitution I

As a first attempt, consider the following:

$$
\begin{aligned}
y\{e/x\} &= \begin{cases} e & \text{if } y = x \\ y & \text{otherwise} \end{cases} \\
(e_1\ e_2)\{e/x\} &= (e_1\{e/x\})\ (e_2\{e/x\}) \\
(\lambda y.e_1)\{e/x\} &= \lambda y.e_1\{e/x\}
\end{aligned}
$$

What's wrong with this definition?

It substitutes bound variables too!

$$(\lambda y.y)\{3/y\} = (\lambda y.3)$$

Substitution II

Okay... since we are working up to α-equivalence, let's replace the variables used in abstractions so they are different than the variable being substituted.

Substitution II

Okay... since we are working up to α-equivalence, let's replace the variables used in abstractions so they are different than the variable being substituted.

$$y\{e/x\} \quad = \quad \begin{cases} e & \text{if } y = x \\ y & \text{otherwise} \end{cases}$$

Substitution II

Okay... since we are working up to α-equivalence, let's replace the variables used in abstractions so they are different than the variable being substituted.

$$
\begin{array}{rcl}
y\{e/x\} & = & \begin{cases} e & \text{if } y = x \\ y & \text{otherwise} \end{cases} \\
(e_1\ e_2)\{e/x\} & = & (e_1\{e/x\})\,(e_2\{e/x\})
\end{array}
$$

Substitution II

Okay... since we are working up to α-equivalence, let's replace the variables used in abstractions so they are different than the variable being substituted.

$$
\begin{aligned}
y\{e/x\} &= \begin{cases} e & \text{if } y = x \\ y & \text{otherwise} \end{cases} \\
(e_1\ e_2)\{e/x\} &= (e_1\{e/x\})\,(e_2\{e/x\}) \\
(\lambda y.e_1)\{e/x\} &= \lambda y.e_1\{e/x\} \qquad\qquad \text{where } y \neq x
\end{aligned}
$$

Substitution II

Okay... since we are working up to α-equivalence, let's replace the variables used in abstractions so they are different than the variable being substituted.

$$
\begin{array}{rcll}
y\{e/x\} & = & \begin{cases} e & \text{if } y = x \\ y & \text{otherwise} \end{cases} & \\
(e_1\ e_2)\{e/x\} & = & (e_1\{e/x\})\,(e_2\{e/x\}) & \\
(\lambda y.e_1)\{e/x\} & = & \lambda y.e_1\{e/x\} & \text{where } y \neq x
\end{array}
$$

What's wrong with this definition?

Substitution II

Okay... since we are working up to α-equivalence, let's replace the variables used in abstractions so they are different than the variable being substituted.

$$
\begin{array}{rcll}
y\{e/x\} & = & \begin{cases} e & \text{if } y = x \\ y & \text{otherwise} \end{cases} & \\
(e_1\ e_2)\{e/x\} & = & (e_1\{e/x\})\,(e_2\{e/x\}) & \\
(\lambda y.e_1)\{e/x\} & = & \lambda y.e_1\{e/x\} & \text{where } y \neq x
\end{array}
$$

What's wrong with this definition?

It leads to variable capture!

$$(\lambda y.x)\{y/x\}$$

Substitution II

Okay... since we are working up to α-equivalence, let's replace the variables used in abstractions so they are different than the variable being substituted.

$$
\begin{aligned}
y\{e/x\} &= \begin{cases} e & \text{if } y = x \\ y & \text{otherwise} \end{cases} \\
(e_1\ e_2)\{e/x\} &= (e_1\{e/x\})\,(e_2\{e/x\}) \\
(\lambda y.e_1)\{e/x\} &= \lambda y.e_1\{e/x\} \qquad \text{where } y \neq x
\end{aligned}
$$

What's wrong with this definition?

It leads to variable capture!

$$(\lambda y.x)\{y/x\} = (\lambda y.y)$$

Substitution III

The correct definition is *capture-avoiding substitution*:

$$
\begin{array}{rcll}
y\{e/x\} & = & \begin{cases} e & \text{if } y \neq x \\ y & \text{otherwise} \end{cases} & \\
(e_1\ e_2)\{e/x\} & = & (e_1\{e/x\})\ (e_2\{e/x\}) & \\
(\lambda y.e_1)\{e/x\} & = & \lambda y.(e_1\{e/x\}) & \text{where } y \neq x \text{ and } y \notin fv(e)
\end{array}
$$