

Elements of Programming Languages

Tutorial 2: Substitution and alpha-equivalence

Solution notes

1. Evaluation

- (a) • $(\lambda x:\text{int}. x) 1$

$$\frac{\overline{\lambda x:\text{int}. x} \Downarrow \overline{\lambda x:\text{int}. x} \quad \overline{1} \Downarrow \overline{1} \quad \overline{1} \Downarrow \overline{1}}{(\lambda x:\text{int}. x) 1 \Downarrow 1}$$

- $(\lambda x:\text{int}. x + 1) 42$

$$\frac{\overline{\lambda x:\text{int}. x + 1} \Downarrow \overline{\lambda x:\text{int}. x + 1} \quad \overline{42} \Downarrow \overline{42} \quad \overline{42 + 1} \Downarrow \overline{43} \quad \overline{1} \Downarrow \overline{1}}{(\lambda x:\text{int}. x + 1) 42 \Downarrow 43}$$

- $(\lambda x:\text{int} \rightarrow \text{int}. x) (\lambda x:\text{int}. x) 1$ Type annotations elided.

$$\frac{\overline{\lambda x. x} \Downarrow \overline{\lambda x. x} \quad \overline{\lambda x. x} \Downarrow \overline{\lambda x. x} \quad \overline{\lambda x. x} \Downarrow \overline{\lambda x. x}}{\overline{(\lambda x. x) (\lambda x. x)} \Downarrow \overline{\lambda x. x}} \quad \overline{1} \Downarrow \overline{1}}{((\lambda x. x) (\lambda x. x)) 1 \Downarrow 1}$$

- $(\star) ((\lambda f:\text{int} \rightarrow \text{int}. \lambda x:\text{int}. f (f x)) (\lambda x:\text{int}. x + 1)) 42$ Type annotations elided.

$$\frac{\overline{(\lambda f. \lambda x. f (f x))} \Downarrow \overline{(\lambda f. \lambda x. f (f x))} \quad \overline{\lambda x. x + 1} \Downarrow \overline{\lambda x. x + 1} \quad \overline{42} \Downarrow \overline{42} \quad \overline{(\lambda x. x + 1)((\lambda x. x + 1)42)} \Downarrow \overline{44}}{\overline{((\lambda f. \lambda x. f (f x)) (\lambda x. x + 1)) 42} \Downarrow \overline{44}}$$

where

$$\frac{\overline{\lambda x. x + 1} \Downarrow \overline{\lambda x. x + 1} \quad \overline{42} \Downarrow \overline{42} \quad \overline{42 + 1} \Downarrow \overline{43}}{\overline{(\lambda x. x + 1)42} \Downarrow \overline{43}} \quad \overline{43 + 1} \Downarrow \overline{44}}{(\lambda x. x + 1)((\lambda x. x + 1)42) \Downarrow 44}$$

- (b) If $e_1 : \tau$ then we can define $\text{let } x = e_1 \text{ in } e_2$ as $(\lambda x:\tau. e_2) e_1$. The evaluation rule for let can be emulated as follows:

$$\frac{e_1 \Downarrow v_1 \quad e_2[v_1/x] \Downarrow v \quad \overline{\lambda x:\tau. e_2} \Downarrow \overline{\lambda x:\tau. e_2} \quad e_1 \Downarrow v_1 \quad e_2[v_1/x] \Downarrow v}{\text{let } x = e_1 \text{ in } e_2 \Downarrow v \implies (\lambda x:\tau. e_2) e_1 \Downarrow v}$$

2. Typechecking

- (a) • $\text{Int} \Rightarrow \text{Int}$

$\{ (x : \text{Int}) \Rightarrow x \}$

- $\text{Int} \Rightarrow \text{Boolean} \Rightarrow \text{Int}$

$\{ (x : \text{Int}) \Rightarrow \{ (y : \text{Boolean}) \Rightarrow x \} \}$

- $(\text{Int} \Rightarrow \text{Boolean} \Rightarrow \text{String}) \Rightarrow (\text{Int} \Rightarrow \text{Boolean}) \Rightarrow (\text{Int} \Rightarrow \text{String})$

```
{(x: Int => Boolean => String) =>
  {(y: Int => Boolean) =>
    {(z: Int) => x(z) (y(z))}}}
```

- (b) • $(\lambda x:\text{int}. x) 1$

$$\frac{\frac{x:\text{int} \vdash x:\text{int}}{\vdash \lambda x:\text{int}. x:\text{int} \rightarrow \text{int}} \quad \frac{}{\vdash 1:\text{int}}}{\vdash (\lambda x:\text{int}. x) 1:\text{int}}$$

- $(\lambda x:\text{int}. x + 1) 42$

$$\frac{\frac{\frac{x:\text{int} \vdash x:\text{int} \quad x:\text{int} \vdash 1:\text{int}}{x:\text{int} \vdash x+1:\text{int}}}{\vdash \lambda x:\text{int}. x+1:\text{int} \rightarrow \text{int}} \quad \frac{}{\vdash 42:\text{int}}}{\vdash (\lambda x:\text{int}. x+1) 42:\text{int}}$$

- $(\lambda x:\text{int} \rightarrow \text{int}. x) (\lambda x:\text{int}. x)$

$$\frac{\frac{\frac{x:\text{int} \rightarrow \text{int} \vdash x:\text{int} \rightarrow \text{int}}{\vdash (\lambda x:\text{int} \rightarrow \text{int}. x):(\text{int} \rightarrow \text{int}) \rightarrow (\text{int} \rightarrow \text{int})} \quad \vdots}{\vdash \lambda x:\text{int} x:\text{int} \rightarrow \text{int}}}{\vdash (\lambda x:\text{int} \rightarrow \text{int}. x) (\lambda x:\text{int}. x):\text{int} \rightarrow \text{int}}$$

- $(\lambda x:\tau. x x)$ **This expression cannot be typed. There is no way to choose τ so that the following derivation can be completed:**

$$\frac{\frac{??}{x:\tau \vdash x:\tau_1 \rightarrow \tau_2} \quad \frac{??}{x:\tau \vdash x:\tau_1}}{x:\tau \vdash x x:\tau_2}}{\vdash \lambda x:\tau. x x:\tau_2}$$

For if $\tau = \tau_1$ then we would also have to have $\tau = \tau_1 \rightarrow \tau_2$, i.e. $\tau_1 = \tau_1 \rightarrow \tau_2$ which is not possible if equality is structural.

3. Alpha-equivalence for L_{Lam}

- (a) The missing rules are:

$$\boxed{e_1 \equiv_{\alpha} e_2}$$

$$\frac{\frac{e \equiv_{\alpha} e' \quad e_1 \equiv_{\alpha} e'_1 \quad e_1 \equiv_{\alpha} e'_1}{\text{if } e \text{ then } e_1 \text{ else } e_2 \equiv_{\alpha} \text{if } e' \text{ then } e'_1 \text{ else } e'_2}}{\frac{e_1(x \leftrightarrow z) \equiv_{\alpha} e_2(y \leftrightarrow z) \quad z \notin FV(e_1, e_2)}{\lambda x. e_1 \equiv_{\alpha} \lambda y. e_2} \quad \frac{e_1 \equiv_{\alpha} e'_1 \quad e_1 \equiv_{\alpha} e'_1}{e_1 e_2 \equiv_{\alpha} e'_1 e'_2}}$$

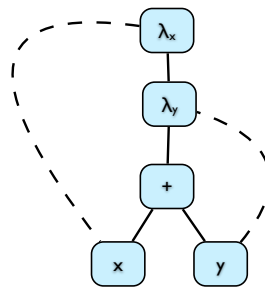
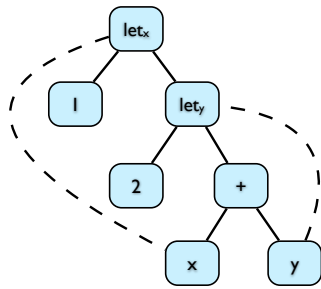
Point this out: To be precise, we should also extend FV as follows:

$$\begin{aligned} FV(\lambda x:\tau. e) &= FV(e) - \{x\} \\ FV(e_1 e_2) &= FV(e_1) \cup FV(e_2) \end{aligned}$$

- (b) Which of the following alpha-equivalence relationships hold?

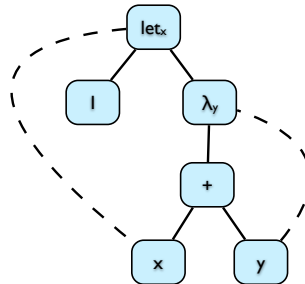
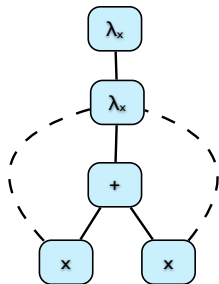
$$\begin{aligned} \text{if true then } y \text{ else } z &\equiv_{\alpha} y && \text{FALSE} \\ \text{let } x = y \text{ in (if } x \text{ then } y \text{ else } z) &\equiv_{\alpha} \text{let } z = y \text{ in (if } x \text{ then } y \text{ else } z) && \text{FALSE} \\ \lambda x. (\text{let } y = x \text{ in } y + y) &\equiv_{\alpha} \lambda x. (\text{let } x = x \text{ in } x + x) && \text{TRUE} \end{aligned}$$

- (c) The pictures should be as follows:



let x = 1 in let y = 2 in x + y

$\lambda x. \lambda y. x + y$



$\lambda x. \lambda x. x + x$

let x = 1 in $\lambda y. x + y$

4. (*) Naive substitution and variable capture

(a)

$$\begin{aligned}
 (\lambda y. \lambda z. ((x + y) + z))[y \times z/x] &= \lambda y. \lambda z. (((y \times z) + y) + z) \\
 (\text{if } x == y \text{ then } \lambda z.x \text{ else } \lambda x.x)[z/x] &= \text{if } z == y \text{ then } \lambda z.z \text{ else } \lambda x.z
 \end{aligned}$$

(b)

$$\begin{aligned}
 \lambda y. \lambda z. ((x + y) + z) &\equiv_{\alpha} \lambda a. \lambda b. ((x + a) + b) \\
 \text{if } x == y \text{ then } \lambda z.x \text{ else } \lambda x.x &\equiv_{\alpha} \text{if } x == y \text{ then } \lambda c.x \text{ else } \lambda d.d
 \end{aligned}$$

(c)

$$\begin{aligned}
 (\lambda a. \lambda b. ((x + a) + b))[y \times z/x] &= \lambda a. \lambda b. (((y \times z) + a) + b) \\
 (\text{if } x == y \text{ then } \lambda c.x \text{ else } \lambda d.d)[z/x] &= \text{if } z == y \text{ then } \lambda c.z \text{ else } \lambda d.d
 \end{aligned}$$

Illustrate that the substitutions performed without α -conversion lead to variable capture, and different binding structure from those performed after α -converting to fresh names.