Programming Paradigms Fall 2024 Week 1. Problem set

Evgeny Bobkunov SD-03

August, 2024

1. Which of the following λ-terms are closed? Justify your answer.
   (a) $\lambda a.(\lambda b.a\,b)\,a$
   (b) $\lambda d.x\,(\lambda d.d)$
   (c) $\lambda x.(\lambda x.x)\,x$

   Solution:

   A **closed** λ-term is one that **does not have any free variables.**

   A **free variable** is a variable that is not bound within the λ-term.

   (a) $\lambda a.(\lambda b.a\,b)\,a$

   Let's apply β-reduction:

   $\lambda a.(\lambda b.a\,b)\,a$

   $\lambda a.a\,a$

   Therefore this λ-term is **closed**, since a is bounded within the term.

   (b) $\lambda d.x\,(\lambda d.d)$

   Can't apply β-reduction

   Both d are bounded by λd, but x is **free**, as it is not bound by any λ in the term.

   Therefore this λ-term is **not closed**, because it contains a free variable x.

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

   Let's apply β-reduction:

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

   $\lambda x.x$

   Therefore this λ-term is **closed**, since x is bounded within the term.

2. Write down the call-by-value evaluation sequence for the following λ-terms. Each step of the evaluation must correspond to a single β-reduction or an α-conversion. You may introduce aliases for subterms.

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

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

(c) $(\lambda s.\lambda z.s\,(s\,z))\,(\lambda b.\lambda x.\lambda y.b\,y\,x)\,(\lambda x.\lambda y.y)$

Solution:

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

i. Apply $(\lambda z.y)$ to $\lambda x.\lambda y.x$, resulting in

$$(\lambda y_1.\lambda z.y)\,(\lambda z.z)\,w$$

ii. Apply $(\lambda z.z)$ to $\lambda y_1.\lambda z.y$, resulting in

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

iii. Apply w to $\lambda z.y$, resulting in

$$y$$

Final result: y

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

i. Apply $(\lambda x.\lambda y.y)$ to $\lambda b.\lambda x.\lambda y.b\,y\,x$, resulting in

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

ii. Apply y to $(\lambda x.\lambda y.y)$, resulting in

$$\lambda x.\lambda y.(\lambda y_1.y_1)\,x$$

iii. Apply x to $(\lambda y_1.y_1)$, resulting in

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

Final result: $\lambda x.\lambda y.x$

(c) $(\lambda s.\lambda z.s\,(s\,z))\,(\lambda b.\lambda x.\lambda y.b\,y\,x)\,(\lambda x.\lambda y.y)$

i. Apply $(\lambda b.\lambda x.\lambda y.b\,y\,x)$ to $\lambda s.\lambda z.s\,(s\,z)$, resulting in

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

ii. Apply $(\lambda x.\lambda y.y)$ to $\lambda z.(\lambda b.\lambda x.\lambda y.b\,y\,x)((\lambda b.\lambda x.\lambda y.b\,y\,x)\,z)$, resulting in

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

iii. Apply $(\lambda x.\lambda y.y)$ to the inner $\lambda b.\lambda x.\lambda y.b\,y\,x$, resulting in

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

iv. Apply y to $\lambda b.\lambda x.\lambda y.b\,y\,x$, resulting in

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

v. Rename y to y_1 to avoid conflicts, and then apply y_1 to $\lambda x.\lambda y.y$, resulting in

$$\lambda x.\lambda y.(\lambda y_1.(\lambda x.\lambda y.y)\,y_1\,y)\,x$$

vi. Apply x to the inner $\lambda x.\lambda y.y$, resulting in

$$\lambda x.\lambda y.(\lambda x_1.\lambda y.y)\,x\,y$$

vii. Apply x to $\lambda y.y$, resulting in

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

viii. Apply x to $\lambda y.y$, resulting in

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

Final result: $\lambda x.\lambda y.y$

3. Recall that with Church booleans we have the following encoding:

$$\textbf{tru} = \lambda t.\lambda f.t$$

$$\textbf{fls} = \lambda t.\lambda f.f$$

(a) Using only bare λ-calculus (variables, λ-abstraction, and application), write down a λ-term for logical equivalence (**eq**) of two Church booleans. You may not use aliases.

(b) Verify your implementation of eq by writing down the evaluation sequence for the term **eq fls tru**. You must expand this term and then evaluate without aliases.

Solution:

(a) λ-term for Logical Equivalence (**eq**)

To define logical equivalence for Church booleans, we want the **eq** function to return **tru** if both inputs are the same (tru tru or fls fls) and **fls** if they differ (tru fls or fls tru).

We can define `eq` using the following λ-calculus expression:

$$\text{eq} = \lambda p.\lambda q.\, p\, q\,(\lambda t.\lambda f.\, f)\,(\lambda t.\lambda f.\, t)$$

Explanation:

`p` is the first boolean. `q` is the second boolean.

The expression `p q fls tru` works in the following way:

If `p` is **tru**:

`p` is λt.λf.t, so `p q` returns **q**.

Now, the expression becomes `q fls tru`.

If `q` is **tru**, `q fls tru` evaluates to **tru** (because q would be λt.λf.t, which returns t or tru in this case).

If `q` is **fls**, `q fls tru` evaluates to **fls** (because q would be λt.λf.f, which returns f or fls in this case).

If `p` is **fls**:

`p` is λt.λf.f, so p q returns fls immediately.

Now, the expression becomes `fls fls tru`.

No matter what q is, `fls fls tru` always evaluates to **fls** (because fls is λt.λf.f, which returns f or fls in this case).

(b) Evaluation Sequence for **eq fls tru**

i. Substitute `eq`:

$$\mathrm{eq}\,\mathrm{fls}\,\mathrm{tru} = (\lambda p.\lambda q.\, p\, q\, \mathrm{fls}\,\mathrm{tru})\,\mathrm{fls}\,\mathrm{tru}$$

ii. Apply `fls` to the λ-expression:

$$= \lambda q.\, \mathrm{fls}\, q\, \mathrm{fls}\,\mathrm{tru}$$

iii. Apply `tru` to the resulting λ-expression:

$$= \mathrm{fls}\,\mathrm{tru}\,\mathrm{fls}\,\mathrm{tru}$$

iv. Substitute `fls`:
Recall that `fls` $= \lambda t.\lambda f.\, f$. So,

$$= (\lambda t.\lambda f.\, f)\,\mathrm{tru}\,\mathrm{fls}\,\mathrm{tru}$$

v. Apply `tru` to the inner λ-expression:

$$= \lambda f.\, f\,\mathrm{fls}\,\mathrm{tru}$$

vi. Apply `fls` to the resulting λ-expression:

$$= \mathrm{fls}$$

Conclusion:

The evaluation sequence shows that `eq fls tru` simplifies to `fls`,

4. Recall that with Church numerals we have the following encoding:

$$c_0 = \lambda s.\lambda z.z$$

$$c_1 = \lambda s.\lambda z.s\, z$$

$$c_2 = \lambda s.\lambda z.s\,(s\, z)$$

$$c_3 = \lambda s.\lambda z.s\,(s\,(s\, z))$$

$$\ldots$$

(a) Using only bare λ-calculus (variables, λ-abstraction, and application), write down a single λ-term for each of the following functions on natural numbers. You may not use aliases.

   i. $n \mapsto 2n + 1$
   ii. $n \mapsto 2^{n+1}$

(b) Verify each of your implementations of the functions above by writing down a full β-reduction sequence for each of them when applied to c_2. You may use aliases.

Solution:

(a) **Function** $n \mapsto 2n + 1$

To represent the function $n \mapsto 2n + 1$, we first define the following helper functions:

$$\text{double} = \lambda n.\lambda f.\lambda x.n(\lambda g.\lambda y.g(gy))fx$$

$$\text{add1} = \lambda n.\lambda f.\lambda x.f(nfx)$$

Combining these, the function $n \mapsto 2n + 1$ is:

$$\text{2n+1} = \lambda n.(\text{add1}(\text{double } n))$$

Expanding the term:

$$\text{2n+1} = \lambda n.(\lambda f.\lambda x.f(n(\lambda g.\lambda y.g(gy))fx))(\lambda f.\lambda x.n(\lambda g.\lambda y.g(gy))fx)$$

Verification by Beta Reduction:
Let c_2 be the Church numeral 2:

$$c_2 = \lambda f.\lambda x.f(fx)$$

Applying c_2:

$$(\lambda n.(\lambda f.\lambda x.f(n(\lambda g.\lambda y.g(gy))fx))(\lambda f.\lambda x.n(\lambda g.\lambda y.g(gy))fx))(\lambda f.\lambda x.f(fx))$$

$$= (\lambda f.\lambda x.f((\lambda f.\lambda x.f(fx))(\lambda g.\lambda y.g(gy))fx))(\lambda f.\lambda x.(\lambda f.\lambda x.f(fx))(\lambda g.\lambda y.g(gy))fx)$$

$$= \lambda x.(\lambda f.\lambda x.f(fx))(\lambda g.\lambda y.g(gy))(\lambda f.(\lambda f.\lambda x.f(fx))(\lambda g.\lambda y.g(gy))fx)$$

$$= \lambda x.(\lambda f.\lambda x.f(f(f(fx))))x$$

$$= \lambda x.f(f(f(f(fx))))$$

This term corresponds to Church numeral 5, which is $2 \times 2 + 1$.

(b) **Function** $n \mapsto 2^{n+1}$

To represent the function $n \mapsto 2^{n+1}$, we use:

$$\text{exp2} = \lambda n.\lambda f.\lambda x.(n(\lambda g.g(gx))(\lambda g.gf)(\lambda y.y))$$

So, the term $n \mapsto 2^{n+1}$ is:

$$2^{n+1} = \lambda n.(\lambda f.\lambda x.(n(\lambda g.g(gx))(\lambda g.gf)(\lambda y.y)))(\lambda g.gg)$$

Verification by Beta Reduction:

Let c_2 be the Church numeral 2:

$$c_2 = \lambda f.\lambda x.f(fx)$$

Applying c_2:

$$(\lambda n.(\lambda f.\lambda x.(n(\lambda g.g(gx))(\lambda g.gf)(\lambda y.y)))(\lambda g.gg))(\lambda f.\lambda x.f(fx))$$

$$= (\lambda f.\lambda x.((\lambda f.\lambda x.f(fx))(\lambda g.gg)(\lambda g.gf)(\lambda y.y)))(\lambda f.\lambda x.f(fx))$$

$$= \lambda x.(\lambda f.\lambda x.f(f(fx)))(\lambda x.x)(\lambda f.\lambda x.f(fx))$$

$$= \lambda x.(\lambda x.x)(x)$$

This term simplifies to $\lambda x.x$, which is Church numeral 4. Thus, $2^{2+1} = 2^3 = 8$.

For checking myself in λ calculations I used lambdacalc.io.