LAMBDA CALCULUS WORKSHEET 1

L. COLLINS

29th June 2026

In our first lesson, we introduced the syntax of the type-free λ-calculus. Terms are built from variables, abstraction, and application:

$$M ::= v \mid (\lambda v.M) \mid (MM), \qquad v \in V.$$

Intuitively, variables get their meanings from an environment, applications apply one meaning to another, and abstractions denote functions. We also use the usual conventions that application associates to the left, the body of a λ extends as far right as possible, and consecutive λs may be compressed.

Recall also that the set of subterms of a λ-term is defined recursively by

$$\begin{aligned} \text{subterms}(v) &:= \{v\}, \\ \text{subterms}(\lambda v.A) &:= \text{subterms}(A) \cup \{\lambda v.A\}, \\ \text{subterms}(AB) &:= \text{subterms}(A) \cup \text{subterms}(B) \cup \{AB\}. \end{aligned}$$

Exercises

1. Meaning of expressions

Describe the informal meaning of each expression.

(a) $\lambda x.y$

(b) $\lambda x.xx$

(c) $\lambda xy.x$

(d) $\lambda xy.y$

(e) $\lambda xyz.xz(yz)$

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

2. Remove as many parentheses as possible

Rewrite each expression using the notational conventions.

(a) $(\lambda x.(xy))$

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

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

(d) $((\lambda x.(xx))(\lambda y.(yy)))$

(e) $(\lambda x.(\lambda y.(\lambda z.((xz)(yz)))))$

(f) $((xy)(z(\lambda x.x)))$

3. Insert all parentheses

Rewrite each expression with the full amount of parentheses.

(a) xyz

(b) $\lambda x.xy$

(c) $\lambda xy.x$

(d) $(\lambda x.x)yz$

(e) $\lambda xyz.xz(yz)$

(f) $x(\lambda y.yz)w$

4. Syntax trees and grammatical derivations

For each of the following expressions, first insert all missing parentheses, and then draw the syntax tree.

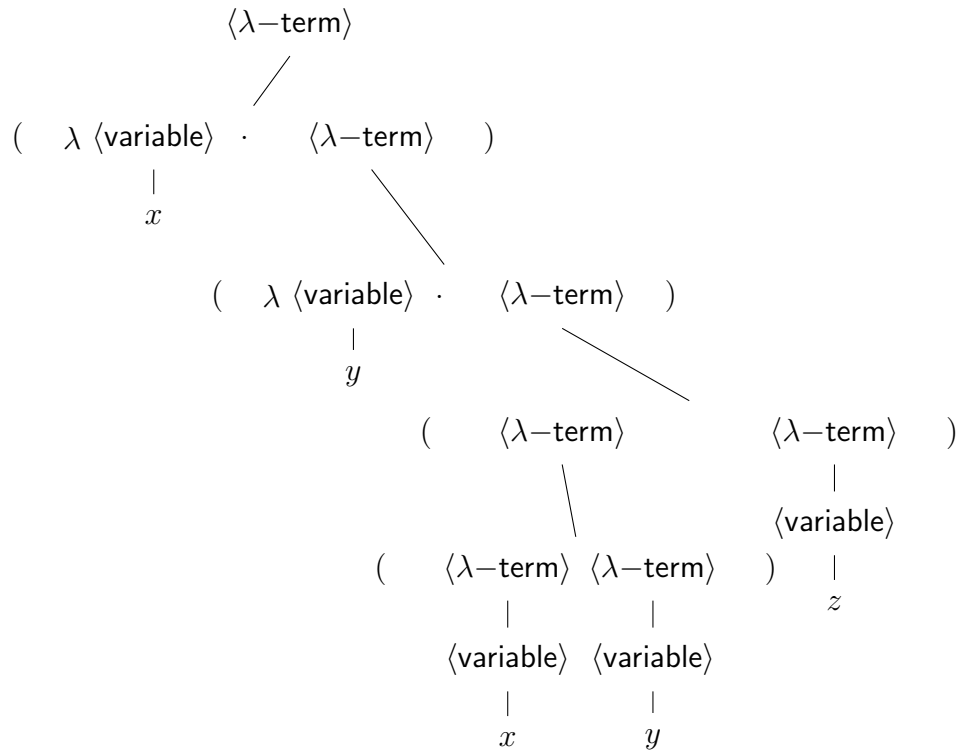
(a) $\lambda x.xy$

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

(c) $\lambda xy.x$

(d) $\lambda xyz.xz(yz)$

For example, $\lambda xy.xyz$ is $(\lambda x.(\lambda y.((xy)z)))$, and its syntax tree is:



5. Valid terms, notation, and first applications

During the lesson, we distinguished between the strict grammar of λ-terms and the informal notational conventions which allow us to omit some parentheses.

We also introduced the idea that an abstraction $\lambda x.A$ behaves like an anonymous function waiting to be applied to an argument.

(a) Which of the following are valid λ-terms according to the strict grammar? For each invalid expression, explain what is wrong.

(i) x
(ii) xy
(iii) $\lambda x.x$
(iv) $(\lambda x.(xy))$
(v) $(\lambda xy.x)$
(vi) $(\lambda x.(\lambda y.x))$
(vii) $x\lambda$
(viii) $((\lambda x.x)y)$

(b) Rewrite the following informal expressions as strict λ-terms, with all parentheses inserted.

(i) $\lambda x.x$
(ii) $\lambda x.xy$
(iii) $\lambda xy.x$
(iv) $(\lambda x.x)y$
(v) xyz
(vi) $\lambda xyz.xz(yz)$

(c) For each of the following expressions, say what the abstraction is waiting to be applied to, and what variable would be replaced.

(i) $(\lambda x.x)y$
(ii) $(\lambda x.xy)z$
(iii) $(\lambda x.xx)(\lambda y.y)$
(iv) $(\lambda y.xy)z$

(d) Using the informal "plug it in" idea from class, simplify the following expressions as far as seems natural. Do not worry yet about giving a formal rule.

(i) $(\lambda x.x)y$
(ii) $(\lambda x.y)z$
(iii) $(\lambda x.xy)z$
(iv) $(\lambda x.xx)y$
(v) $(\lambda y.xy)z$
(vi) $(\lambda x.(\lambda y.x))z$

6. Subterms

Find all subterms of each expression. Do the full derivation, from the definition of subterms(A), as we did during the lesson.

(a) xy

(b) $\lambda x.xy$

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

(d) $\lambda xy.x$

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

(f) $\lambda xyz.xz(yz)$

7. Another recursive definition

We define the *size* of a λ-term recursively as follows:

$$\begin{aligned} \text{size}(v) &:= 1, \\ \text{size}(\lambda v.A) &:= 1 + \text{size}(A), \\ \text{size}(AB) &:= \text{size}(A) + \text{size}(B). \end{aligned}$$

Work out the size of each of the λ-terms from the previous exercise.

Luke Collins B.Sc. (*Melit.*), M.Sc. (*Warw.*)
Email address: luke.collins.22@ucl.ac.uk
Personal webpage: https://lc.mt

Room 713
Department of Mathematics
University College London
25 Gordon Street, London, UK

LAMBDA CALCULUS WORKSHEET 2

L. COLLINS

2nd July 2026

In the last worksheet we introduced the syntax of the type-free λ-calculus. This week we study variables more carefully. A variable occurrence may be *free* or *bound*, and this distinction is essential when we substitute one term into another. If we substitute carelessly, we may accidentally turn a free variable into a bound one. This is called *variable capture*.

Recall that the free variables of a term are defined recursively by

$$\begin{aligned} \mathrm{fv}(v) &:= \{v\}, \\ \mathrm{fv}(\lambda v.A) &:= \mathrm{fv}(A) \setminus \{v\}, \\ \mathrm{fv}(AB) &:= \mathrm{fv}(A) \cup \mathrm{fv}(B). \end{aligned}$$

An occurrence of a variable v is *bound* if it lies in the scope of some λv; otherwise it is *free*. A closed term is a term with no free variables.

Substitution must avoid variable capture. For any terms A, B and variable v, $A[v \leftarrow B]$ (also sometimes written $A[v := B]$) means: substitute B for every *free* occurrence of v in A. The recursive definition is:

$$\begin{aligned} u[v \leftarrow B] &:= \begin{cases} u, & \text{if } u \neq v, \\ B, & \text{if } u = v, \end{cases} \\ (AC)[v \leftarrow B] &:= A[v \leftarrow B]\, C[v \leftarrow B], \\ (\lambda u.A)[v \leftarrow B] &:= \begin{cases} \lambda u.A, & \text{if } u = v, \\ \lambda u.(A[v \leftarrow B]), & \text{if } u \neq v \text{ and } u \notin \mathrm{fv}(B), \\ \lambda z.(A[u \leftarrow z][v \leftarrow B]), & \text{if } u \neq v \text{ and } u \in \mathrm{fv}(B), \end{cases} \end{aligned}$$

where in the last case z is chosen fresh, i.e., so that it does not occur in A or B.

Finally, α-reduction is simply the renaming of bound variables:

$$\lambda v.A \to_\alpha \lambda v'.A[v \leftarrow v'] \qquad \text{provided } v' \notin \mathrm{fv}(A).$$

So $\lambda y.xy \to_\alpha \lambda z.xz$, but $\lambda y.xy \not\to_\alpha \lambda x.xx$.

Exercises

1. Subterms and trees

Find all subterms of each expression. Then draw the tree of the term, using δ for application nodes.

(a) $(\lambda p.pq)(rs)$

(b) $\lambda a.(ba)(\lambda c.ac)$

(c) $(\lambda m.\lambda n.mn)(\lambda r.sr)$

(d) $u((\lambda t.tt)v)$

For example, if a term is an application AB, then its tree has a δ at the root, with the tree of A on the left and the tree of B on the right. Subterms are precisely the terms corresponding to subtrees.

2. Free variables

Compute the set of free variables of each expression, showing your working from the definition. Decide also whether the expression is closed.

(a) $\lambda a.ab$

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

(c) $(\lambda f.\lambda x.fx)(\lambda y.cy)$

(d) $\lambda u.(\lambda v.uv)(wu)$

(e) $\lambda r.\lambda s.r(st)$

(f) $(\lambda k.k)(\lambda k.\ell k)$

(g) $\lambda h.h(\lambda h.gh)$

(h) $(\lambda q.qr)(\lambda r.qr)$

3. Free and bound occurrences

In each expression, label each occurrence of a variable. For each occurrence, say whether it is free or bound. If it is bound, say which λ binds it.

(a) $(\lambda p.qp)(\lambda q.pq)$

(b) $\lambda m.n(\lambda n.mn)n$

(c) $(\lambda a.\lambda b.ab)c$

(d) $\lambda z.(\lambda x.zx)(xz)$

(e) $\lambda f.(\lambda x.f(xx))(\lambda f.fx)$

For instance, in $(\lambda p.p)q$, the occurrence of p in the body is bound by λp, while the occurrence of q is free.

4. Binders in ordinary mathematics

The λ is a variable binder. This is not so different from the dummy variables which occur in integrals and summations.

(a) In the expression

$$\int_0^1 (x+y)\,dx,$$

which occurrences are bound, and which symbols are free? Which of the following relabellings are legitimate?

$$\int_0^1 (x+y)\,dx \quad \rightsquigarrow \quad \int_0^1 (t+y)\,dt,$$

$$\int_0^1 (x+y)\,dx \quad \rightsquigarrow \quad \int_0^1 (y+y)\,dy.$$

(b) Explain why

$$\int_0^1 xy\,dx = \int_0^1 ty\,dt$$

is a harmless change of dummy variable, but

$$\int_0^1 xy\,dx \quad \rightsquigarrow \quad \int_0^1 yy\,dy$$

is not a harmless change of dummy variable.

(c) In the expression

$$\sum_{i=0}^{N}(a_i + j),$$

which symbol is bound by the summation sign? Which symbols are free? Which of the following changes of dummy variable are harmless?

$$\sum_{i=0}^{N}(a_i + j) \quad \rightsquigarrow \quad \sum_{k=0}^{N}(a_k + j),$$

$$\sum_{i=0}^{N}(a_i + j) \quad \rightsquigarrow \quad \sum_{j=0}^{N}(a_j + j).$$

(d) Translate the previous two examples into the language of λ-terms. Which of these are legal α-renamings?

$$\lambda x.xy \to_\alpha \lambda t.ty, \qquad \lambda x.xy \to_\alpha \lambda y.yy,$$

$$\lambda i.a_i j \to_\alpha \lambda k.a_k j, \qquad \lambda i.a_i j \to_\alpha \lambda j.a_j j.$$

5. Substitution without capture

Compute the following substitutions. If you need to rename a bound variable first, say so explicitly.

(a) $a[a \leftarrow \lambda t.t]$

(b) $b[a \leftarrow \lambda t.t]$

(c) $(ab)[a \leftarrow \lambda t.rt]$

(d) $(\lambda a.ab)[a \leftarrow c]$

(e) $(\lambda b.ab)[a \leftarrow c]$

(f) $(\lambda b.ab)[a \leftarrow b]$

(g) $(\lambda c.a(\lambda b.cb))[a \leftarrow bc]$

(h) $(\lambda b.\lambda c.abc)[a \leftarrow bc]$

(i) $(\lambda x.x(\lambda y.z)x)[x \leftarrow \lambda z.\lambda y.z]$

(j) $(\lambda x.\lambda y.z)[y \leftarrow zx]$

(k) $(\lambda y.(\lambda y.xy)y)[x \leftarrow y]$

(l) $(\lambda z.x(\lambda x.zx))[x \leftarrow zy]$

(m) $(\lambda a.\lambda b.c(a(\lambda c.bcx)))[c \leftarrow ab]$

(n) $(\lambda xy.w(\lambda y.zwy))[z \leftarrow \lambda xy.w(\lambda y.zwy)]$

6. Syntactic identity, revised

We now identify terms which differ only in the names of their bound variables. For each pair of terms, say whether they are syntactically identical under this revised convention.

(a) u and u'

(b) $\lambda p.p$ and $\lambda q.q$

(c) $\lambda p.qp$ and $\lambda r.qr$

(d) $\lambda p.qp$ and $\lambda q.qq$

(e) $\lambda p.\lambda q.pq$ and $\lambda q.\lambda p.qp$

(f) $\lambda p.\lambda q.pq$ and $\lambda q.\lambda p.pq$

(g) $(\lambda a.ab)c$ and $(\lambda r.rb)c$

(h) $(\lambda a.ab)c$ and $(\lambda r.ra)c$

7. Alpha-reduction

Recall the rule:

$$\lambda v.A \to_\alpha \lambda w.A[v \leftarrow w], \qquad \text{provided } w \notin \mathrm{fv}(A).$$

The side condition is what prevents variable capture.

(a) For each term, perform one or more α-reductions so that no bound variable is called x.

(i) $(\lambda x.px)q$

(ii) $\lambda z.(\lambda x.zx)x$

(iii) $(\lambda x.\lambda y.x(yu))(\lambda u.vu)$

(iv) $\lambda r.(\lambda x.rx)(\lambda x.sx)$

(b) Decide whether each proposed α-reduction is valid. If it is invalid, explain which free variable would be captured.

$$\lambda a.ba \to_\alpha \lambda c.bc, \qquad \lambda a.ba \to_\alpha \lambda b.bb.$$

$$\lambda r.\lambda s.rs \to_\alpha \lambda s.\lambda r.sr, \qquad \lambda r.cr \to_\alpha \lambda c.cc.$$

8. Bound variables and shadowing

We have defined $\mathrm{fv}(A)$, the set of free variables of A. It is also useful to record which variables are used as *binders*. Define $\mathrm{bv}(A)$ recursively by

$$\begin{aligned} \mathrm{bv}(v) &:= \emptyset, \\ \mathrm{bv}(AB) &:= \mathrm{bv}(A) \cup \mathrm{bv}(B), \\ \mathrm{bv}(\lambda v.A) &:= \mathrm{bv}(A) \cup \{v\}. \end{aligned}$$

Thus $\mathrm{bv}(A)$ is the set of variables which occur immediately after a λ in A.

A variable v is *shadowed* when a binder λv occurs inside the scope of another binder λv, such as x in $\lambda x.yz\lambda xy.xz$. To define this formally, let $\mathrm{sh}_S(A)$ be the set of variables shadowed in A, assuming that the variables in S are already being used as binders outside A. Complete the following recursive definition:

$$\begin{aligned} \mathrm{sh}_S(v) &:= \underline{\qquad\qquad\qquad}, \\ \mathrm{sh}_S(AB) &:= \underline{\qquad\qquad\qquad\qquad\qquad}, \\ \mathrm{sh}_S(\lambda v.A) &:= \begin{cases} \{v\} \cup \mathrm{sh}_{S\cup\{v\}}(A), & \text{if } v \in S, \\ \underline{\qquad\qquad\qquad\qquad}, & \text{if } v \notin S. \end{cases} \end{aligned}$$

Finally, define $\mathrm{sh}(A) := \mathrm{sh}_\emptyset(A)$.

For each of the following terms, compute $\mathrm{fv}(A)$, $\mathrm{bv}(A)$, and $\mathrm{sh}(A)$.

(a) $\lambda x.x$

(b) $\lambda x.y$

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

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

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

(f) $\lambda a.(\lambda b.ab)(\lambda a.ba)$

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

(h) $\lambda x.(\lambda y.x(\lambda x.xy))(\lambda y.yx)$

Now α-convert each term in which shadowing occurs so that no variable is shadowed. For example,

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

Luke Collins B.Sc. (*Melit.*), M.Sc. (*Warw.*)
Email address: luke.collins.22@ucl.ac.uk
Personal webpage: https://lc.mt

Room 713
Department of Mathematics
University College London
25 Gordon Street, London, UK

LAMBDA CALCULUS WORKSHEET 3

L. COLLINS

7th July 2026

In the previous worksheet we studied free variables, bound variables, substitution, and α-conversion. This week we treat λ-terms as a computational system: computation is *rewriting*, replacing a special subterm by a simpler one according to a fixed rule.

The three reduction axioms are:

$$\begin{array}{lrll} (\alpha) & \lambda v.A & \to_\alpha & \lambda w.A[v \leftarrow w] \quad \text{if } w \notin \mathrm{fv}(A), \\ (\beta) & (\lambda v.A)B & \to_\beta & A[v \leftarrow B], \\ (\eta) & \lambda v.Av & \to_\eta & A \quad \text{if } v \notin \mathrm{fv}(A). \end{array}$$

The left-hand side of one of these rules is called a *redex*. Thus $(\lambda v.A)B$ is a β-redex, and $\lambda v.Av$ is an η-redex, provided the side condition is satisfied.

These reductions may also happen inside a larger term. For $r \in \{\alpha, \beta, \eta\}$, the relation $\to_r$ is compatible with application and abstraction:

$$\frac{A \to_r B}{AC \to_r BC} \qquad \frac{A \to_r B}{CA \to_r CB} \qquad \frac{A \to_r B}{\lambda x.A \to_r \lambda x.B}.$$

These are the *compatibility rules*: they say that a reduction may take place in the left side of an application, the right side of an application, or inside the body of an abstraction.

Finally, $\twoheadrightarrow_r$ denotes zero or more r-steps; in particular $A \twoheadrightarrow_r A$, and chains of such steps may be joined together.

A term is in *β-normal form* if it contains no subterm of the form $(\lambda v.A)B$, and in *η-normal form* if it contains no subterm of the form $\lambda v.Av$ with $v \notin \mathrm{fv}(A)$. It is in *$\beta\eta$-normal form* exactly when neither a β- nor an η-reduction step is possible.

Exercises

1. Beta-redexes and one-step beta-reduction

In each term, underline every β-redex. For each redex, write the corresponding one-step β-reduction. Do not reduce further than one step.

E.g. In $(\lambda x.(\lambda y.xy)x)z$, there are two β-redexes:

$$\underline{(\lambda x.\underline{(\lambda y.xy)x})z},$$

and thus there are two distinct one-step β-reductions, namely

$$(\lambda x.\underline{(\lambda y.xy)x})z \to_\beta (\lambda x.xx)z \qquad \text{and} \qquad \underline{(\lambda x.(\lambda y.xy)x)z} \to_\beta (\lambda y.zy)z.$$

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

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

(c) $((\lambda x.x)u)((\lambda y.y)v)$

(d) $\lambda z.(\lambda x.xz)y$

(e) $(\lambda f.fz)((\lambda w.w)u)$

(f) $(\lambda x.(\lambda y.y)x)((\lambda z.z)w)$

(g) $\lambda a.((\lambda b.ba)c)(\lambda d.d)$

(h) $((\lambda m.\lambda n.mn)p)((\lambda q.q)r)$

(i) $(\lambda x.\lambda y.xy)z$

(j) $\lambda z.(\lambda x.xz)(\lambda y.y)$

(k) $(\lambda x.x)((\lambda y.yy)z)$

(l) $(\lambda f.f(fa))(\lambda x.x)$

(m) $(\lambda x.\lambda y.x)(\lambda z.zz)w$

2. Multi-step beta-reduction

Reduce each term as far as possible using β-reduction. In each line, write only one β-step, and underline the redex you are reducing. Remember that substitution sometimes requires α-conversion to avoid capture.

E.g. For $(\lambda x.\lambda y.y)(\lambda z.z)u$, one possible sequence is

$$\begin{aligned} &\underline{(\lambda x.\lambda y.y)(\lambda z.z)}u \\ \to_\beta\ &\underline{(\lambda y.y)u} \\ \to_\beta\ &u \end{aligned}$$

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

(b) $(\lambda f.\lambda x.fx)(\lambda u.u)v$

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

(d) $(\lambda x.\lambda y.xy)(\lambda z.yz)$

(e) $(\lambda x.\lambda y.yx)((\lambda z.z)u)v$

(f) $(\lambda x.\lambda y.x(yx))(\lambda z.yz)w$

(g) $(\lambda x.\lambda z.x(\lambda y.zy))(zy)$

(h) $(\lambda x.\lambda y.\lambda z.xz(yz))y(\lambda s.s)w$

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

(j) $(\lambda g.\lambda x.g(gx))(\lambda y.(\lambda z.z)y)a$

(k) $(\lambda x.(\lambda y.yx)((\lambda z.z)x))((\lambda w.w)u)$

3. Reductions in context

The compatibility rules say that a reduction may take place inside a larger term. It is often helpful to write a term as a context with a hole, such as $C[\Box]$, and then reduce the subterm which fills the hole.

For each proposed one-step reduction below, decide whether it is valid. If it is valid, write the redex R, the reduct R', and a context $C[\Box]$ such that the reduction has the form

$$C[R] \to_\beta C[R'].$$

If it is invalid, explain briefly what has gone wrong, and give one valid one-step β-reduction starting from the same term.

E.g. In $z((\lambda x.x)q)$, the reduction happens inside the context $C[\Box] = z(\Box)$. Here $R = (\lambda x.x)q$ and $R' = q$, so

$$z((\lambda x.x)q) = C[R] \to_\beta C[R'] = zq.$$

(a) $h((\lambda x.fx)((\lambda y.y)a)) \to_\beta h(f((\lambda y.y)a))$

(b) $h((\lambda x.fx)((\lambda y.y)a)) \to_\beta h(fa)$

(c) $(\lambda x.\lambda y.yx)((\lambda z.z)u) \to_\beta \lambda y.y((\lambda z.z)u)$

(d) $(\lambda x.\lambda y.yx)((\lambda z.z)u) \to_\beta \lambda y.yu$

(e) $\lambda t.((\lambda x.xt)s)((\lambda y.y)t) \to_\beta \lambda t.(st)((\lambda y.y)t)$

(f) $\lambda t.((\lambda x.xt)s)((\lambda y.y)t) \to_\beta \lambda t.(st)t$

(g) $(\lambda x.\lambda y.x)y \to_\beta \lambda y.y$

(h) $(\lambda x.\lambda y.x)y \to_\beta \lambda z.y$

(i) $((\lambda f.f(fa))(\lambda x.x))b \to_\beta ((\lambda x.x)((\lambda x.x)a))b$

(j) $((\lambda f.f(fa))(\lambda x.x))b \to_\beta ab$

4. Eta-reduction

Recall the rule:

$$\lambda v.Av \to_\eta A \qquad \text{provided } v \notin \text{fv}(A).$$

This rule removes a function which merely passes its argument on. The side condition matters: $\lambda x.(fx)x$ is not an η-redex, because it has the form $\lambda x.Ax$ only with $A = fx$, and $x \in \text{fv}(fx)$.

For each term below, underline all valid η-redexes and perform one η-reduction in each possible position. If a subterm looks like an η-redex but fails the side condition, say why.

E.g. In $\lambda x.(\lambda y.hy)x$ there are two possible one-step η-reductions:

$$\underline{\lambda x.(\lambda y.hy)x} \to_\eta \lambda y.hy \qquad \text{and} \qquad \lambda x.(\underline{\lambda y.hy})x \to_\eta \lambda x.hx.$$

(a) $\lambda x.xfx$

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

(c) $(\lambda x.fx)(\lambda y.gy)$

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

(e) $\lambda x.(g(\lambda y.hy))x$

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

(g) $\lambda z.((\lambda x.fx)z)$

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

(i) $(\lambda f.\lambda x.fx)(\lambda y.hy)$

5. Mixed reductions

Reduce each term to $\beta\eta$-normal form. In each line, write only one reduction step. You may use α-conversion whenever it is needed to avoid capture or to make the expression clearer.

E.g. For $\lambda z.(\lambda y.gy)z$, first apply β-reduction and then η-reduction:

$$\begin{aligned} & \lambda z.\underline{(\lambda y.gy)z} \\ \to_\beta\ & \underline{\lambda z.gz} \\ \to_\eta\ & g \end{aligned}$$

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

(b) $(\lambda f.\lambda x.fx)g$

(c) $(\lambda f.\lambda x.f(xx))(\lambda y.y)$

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

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

(f) $\lambda u.(\lambda x.hx)((\lambda z.z)u)$

(g) $(\lambda x.\lambda y.x)(\lambda z.zz)a$

(h) $(\lambda f.\lambda x.f(fx))(\lambda y.gy)$

(i) $(\lambda x.(\lambda y.y)x)(\lambda z.kz)$

(j) $\lambda x.((\lambda f.fx)(\lambda y.gy))$

(k) $(\lambda f.\lambda x.f(fx))(\lambda y.hy)a$

(l) $(\lambda x.\lambda y.\lambda z.xz(yz))(\lambda t.t)(\lambda s.ks)w$

6. The looping term

Define

$$\Omega := (\lambda x.xx)(\lambda x.xx).$$

(a) Perform one β-reduction step on Ω.

(b) Does Ω have a β-normal form? Give a brief justification for your answer.

(c) Let $M := (\lambda x.a)\Omega$. Find two different β-reduction sequences: one finite, one infinite.

7. Boolean logic

Define

$$\mathsf{True} := \lambda a.\lambda b.a, \qquad \mathsf{False} := \lambda a.\lambda b.b, \qquad \mathsf{If} := \lambda x.x.$$

Thus a Boolean is a term which chooses one of two arguments.

(a) Show that $\mathsf{If}\ \mathsf{True}\ a\ b \twoheadrightarrow_\beta a$.

(b) Show that $\mathsf{If}\ \mathsf{False}\ a\ b \twoheadrightarrow_\beta b$.

(c) Define

$$\mathsf{And} := \lambda p.\lambda q.pq\,\mathsf{False}.$$

Verify that

$$\mathsf{And}\ \mathsf{True}\ \mathsf{True} \twoheadrightarrow_\beta \mathsf{True} \qquad \text{and} \qquad \mathsf{And}\ \mathsf{True}\ \mathsf{False} \twoheadrightarrow_\beta \mathsf{False}.$$

(d) Define

$$\mathsf{Or} := \lambda p.\lambda q.p\,\mathsf{True}\,q, \qquad \mathsf{Not} := \lambda p.p\,\mathsf{False}\,\mathsf{True}.$$

Verify that

$$\mathsf{Or}\ \mathsf{False}\ \mathsf{True} \twoheadrightarrow_\beta \mathsf{True} \qquad \text{and} \qquad \mathsf{Not}\ \mathsf{False} \twoheadrightarrow_\beta \mathsf{True}.$$

(e) Reduce

$$\mathsf{If}\ (\mathsf{And}\ \mathsf{True}\ (\mathsf{Not}\ \mathsf{False}))\ u\ v$$

as far as possible.

(f) Define a term Xor such that $\mathsf{Xor}\ p\ q$ reduces to True exactly when one of p, q is True and the other is False. Check all four possible inputs.

8. Rock, paper, scissors

We can encode the three moves as three-way selectors:

$$\mathsf{Rock} \coloneqq \lambda r.\lambda p.\lambda s.r, \qquad \mathsf{Paper} \coloneqq \lambda r.\lambda p.\lambda s.p, \qquad \mathsf{Scissors} \coloneqq \lambda r.\lambda p.\lambda s.s.$$

(a) Define a term Beats such that

$$\begin{aligned} \mathsf{Beats}\ \mathsf{Rock} &\twoheadrightarrow_\beta \mathsf{Scissors}, \\ \mathsf{Beats}\ \mathsf{Paper} &\twoheadrightarrow_\beta \mathsf{Rock}, \\ \mathsf{Beats}\ \mathsf{Scissors} &\twoheadrightarrow_\beta \mathsf{Paper}. \end{aligned}$$

(b) Define a term $\mathsf{LosesTo}$ such that

$$\begin{aligned} \mathsf{LosesTo}\ \mathsf{Rock} &\twoheadrightarrow_\beta \mathsf{Paper}, \\ \mathsf{LosesTo}\ \mathsf{Paper} &\twoheadrightarrow_\beta \mathsf{Scissors}, \\ \mathsf{LosesTo}\ \mathsf{Scissors} &\twoheadrightarrow_\beta \mathsf{Rock}. \end{aligned}$$

(c) Using the Booleans from Exercise 7, define a term Wins such that $\mathsf{Wins}\ m\ n$ reduces to True exactly when m beats n. Check:

$$\mathsf{Wins}\ \mathsf{Rock}\ \mathsf{Scissors} \twoheadrightarrow_\beta \mathsf{True}, \qquad \mathsf{Wins}\ \mathsf{Rock}\ \mathsf{Paper} \twoheadrightarrow_\beta \mathsf{False}.$$

(d) Define a term Winner such that $\mathsf{Winner}\ m\ n$ returns the winning move, and returns m in a draw. Check:

$$\begin{aligned} \mathsf{Winner}\ \mathsf{Rock}\ \mathsf{Scissors} &\twoheadrightarrow_\beta \mathsf{Rock}, \\ \mathsf{Winner}\ \mathsf{Rock}\ \mathsf{Paper} &\twoheadrightarrow_\beta \mathsf{Paper}, \\ \mathsf{Winner}\ \mathsf{Paper}\ \mathsf{Paper} &\twoheadrightarrow_\beta \mathsf{Paper}. \end{aligned}$$

Luke Collins B.Sc. (*Melit.*), M.Sc. (*Warw.*)
Email address: luke.collins.22@ucl.ac.uk
Personal webpage: https://lc.mt

Room 713
Department of Mathematics
University College London
25 Gordon Street, London, UK