INTRODUCTION TO LAMBDA-CALCULUS

ANDRÁS SIMON

1. SYNTAX

Throughout, we assume that there's a countable set of variables and, possibly, a set of constants (disjoint from the set of variables) available. We refer to them collectively as *atoms*.

Definition 1.1. A(n abstract) *λ-term* is a finite binary tree such that each leaf is labeled by an atom, every vertex with one child is labeled by λx for some variable x, and vertices with two children are labeled with `apply`.

The above defines the *abstract syntax* of the λ-calculus; this is mathematically cleaner and is simpler to manipulate and to reason about but harder to write than the *concrete syntax* that follows:

Definition 1.2. The set of (concrete) λ-terms is the smallest set X that contains all atoms, and such that $(PQ) \in X$ and $(\lambda x.P) \in X$ for all $P, Q \in X$ and variable x. We call (PQ) an *application* and $(\lambda x.P)$ an *abstraction* (or *λ-abstraction*).

Some examples of λ-terms are: $((\lambda x.x)y)$, $(\lambda x.(\lambda y.((xy)x)))$.

From now on, we'll omit the outermost pair of parenthesis; agree that application associates to the left (so that we may write PQR for $(PQ)R$), abstraction associates to the right, so we write $\lambda x.\lambda y.P$ for $\lambda x.(\lambda y.P)$, except that we further abbreviate this to $\lambda xy.P$. One last convention is that application binds stronger than abstraction, so $\lambda x.PQ$ is $\lambda x.(PQ)$, not $(\lambda x.P)Q$. With these conventions in place, $(\lambda x.(\lambda y.((xy)x)))$ becomes $\lambda xy.xyx$.

The correspondence of the two kinds of λ-terms is illustrated by some examples in Fig. 1. Setting up a formal correspondence between these two is left as an (important) exercise.

The next definition is a good example of what we mean by the abstract version being mathematically simpler.

Definition 1.3. A *subterm* of a term P is a subtree of P. An *occurrence* of a subterm in P is the path leading to it. For a variable x, an occurrence of λx is a vertex labeled by λx. The scope of this occurrence is the subtree below that vertex.

Definition 1.4. An occurrence of a variable x is *bound* in a term if it is in the scope of a λx and *free* otherwise.

If x has at least one bound occurrence in P, we call x a bound variable of P. If x has at least one free occurrence in P, we call x a free variable of P. The set of all free variables of P is denoted by $FV(P)$. A term with no free variables is *closed* (also sometimes called a *combinator*).

Date: May 9, 2012.

Please send corrections, questions, etc. to `asimon@math.bme.hu`.

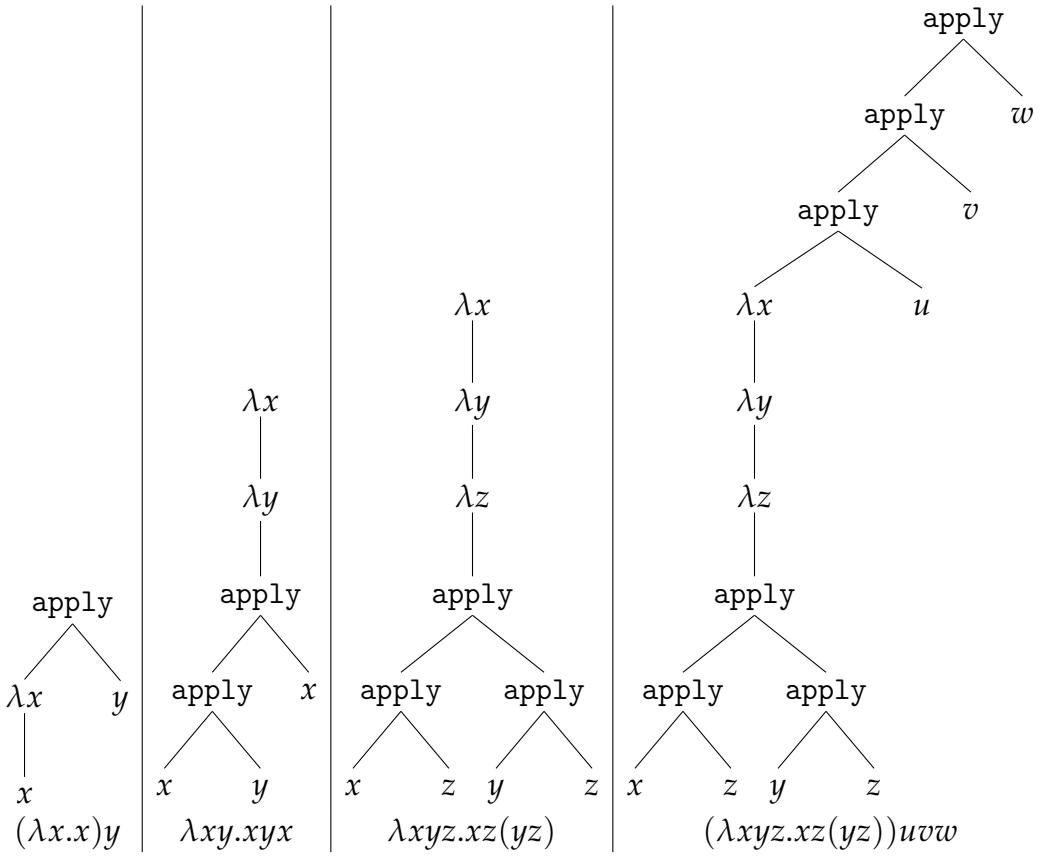
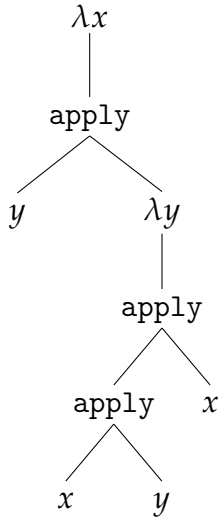


FIGURE 1. λ-terms in tree and concrete form

For example, in the term $\lambda x.y(\lambda y.xyx)$, that is,



y has two occurrences, the first of which is free, and the second bound (by the only occurrence of λy).

Since in practice, we'll be dealing with concrete λ-terms, it's useful to define FV (and the rest of our notions dealing with terms) directly for these.

Proposition 1.5. (i) $FV(x) = \{x\}$ *for a variable* x
(ii) $FV(a) = \emptyset$ *for any other atom* a
(iii) $FV(PQ) = FV(P) \cup FV(Q)$
(iv) $FV(\lambda x.P) = FV(P) \setminus \{x\}$

Until we introduce the variable convention on page 6 below, $\equiv$ means syntactic identity: $M \equiv N$ iff they are the same term.

Definition 1.6 (Substitution)**.** $[N/x]M$ (the result of substituting N for the free occurrences of the variable x in M) is defined as follows:

(i) If M is an atom a, then:
  (a) $[N/x]M \equiv M$ if $a \not\equiv x$
  (b) $[N/x]M \equiv N$ if $a \equiv x$
(ii) If M is an application PQ, then $[N/x]M \equiv ([N/x]P)([N/x]Q)$
(iii) If M is an abstraction $\lambda x.P$, then $[N/x]M \equiv M$
(iv) If M is an abstraction $\lambda y.P$ with $y \not\equiv x$, then:
  (a) $[N/x]M \equiv M$ if $x \notin FV(P)$
  (b) $[N/x]M \equiv \lambda y.[N/x]P$ if $x \in FV(P)$ and $y \notin FV(N)$
  (c) $[N/x]M \equiv \lambda z.[N/x][z/y]P$ if $x \in FV(P)$ and $y \in FV(N)$
  where $y \not\equiv x$ and z is the first variable not in $FV(NP)$.

The extra contortions introduced in the last clause are needed to avoid y being accidentally bound in the result of the substitution. Here's a simple example that shows why this would be a problem. Consider the term $\lambda y.x$; we will see later that it represents the constant function whose value is x. So if we substitute any term N for x, then the result should be a term that represents the constant function whose value is N; and this is indeed the case as long as $y \notin FV(N)$. For example,

$$[z/x]\lambda y.x \equiv \lambda y.[z/x]x \equiv \lambda y.z,$$

as expected. But if we now take y for N, then the naïve way would get us

$$[y/x]\lambda y.x \equiv \lambda y.[y/x]x \equiv \lambda y.y,$$

the identity function! Whereas with our contorted last clause we get the correct

$$[y/x]\lambda y.x \equiv \lambda z.[y/x][z/y]x \equiv \lambda z.[y/x]x \equiv \lambda z.y,$$

that is, a term that represents the constant function whose value is y.

(Even without referring to what functions λ-terms represent, it's clear, that the naïve way of substituting y for x leads to a counterintuitive result: since $FV(\lambda y.x) = \{\, x \,\}$, we expect $FV([y/x]\lambda y.x)$ to be $\{\, y \,\}$, and not $FV(\lambda y.y) = \emptyset$.)

Incidentally, this problem is not specific to λ-calculus: it occurs everywhere (not just in mathematics, but in computer programming, too) where there are operators for binding variables. To mention just one example:

$$\int_0^1 x \, \mathrm{d}y = xy\Big|_{y=0}^{1} = x$$

but substituting x naïvely for y we get

$$\int_0^1 y \, \mathrm{d}y = y^2/2\Big|_{y=0}^{1} = 1/2$$

and not y.

Examples 1.7.
- $[(uv)/x](\lambda x.zy) \equiv \lambda x.zy$
- $[(yu)/y](\lambda x.zy) \equiv \lambda x.z(yu)$
- $[(uv)/x](\lambda y.x(\lambda w.vwx)) \equiv \lambda y.uv(\lambda w.vw(uv))$
- $[w/x](\lambda y.x(\lambda w.vwx)) \equiv \lambda y.w(\lambda z.vzw)$
- $[(\lambda y.xy)/x](\lambda y.x(\lambda x.x)) \equiv \lambda y.(\lambda y.xy)(\lambda x.x)$
- $[(\lambda y.vy)/x](y(\lambda v.xv)) \equiv y(\lambda z.(\lambda y.vy)z)$

Definition 1.8 (Length of terms)**.** $lg(M)$, the *length* of the term M is defined inductively as follows:

(i) $lg(a) = 1$ if a is an atom
(ii) $lg(MN) = lg(M) + lg(N)$
(iii) $lg(\lambda x.M) = 1 + lg(M)$.

Proposition 1.9. (i) $[x/x]M \equiv M$
(ii) *If* $x \notin FV(M)$, *then* $[N/x]M \equiv M$
(iii) $x \in FV(M) \implies FV([N/x]M) = FV(N) \cup (FV(M) \setminus \{x\})$
(iv) $lg([y/x]M) = lg(M)$

Proof. (ii): If M is an atom, then $M \not\equiv x$ by 1.5(i), so $[N/x]M \equiv M$ by 1.6(i)a.

If $M \equiv PQ$, then, since $x \notin FV(P) \cup FV(Q)$ by 1.5(iii), $[N/x](PQ) \equiv [N/x]P[N/x]Q \equiv PQ$ by 1.6(ii) and the induction hypothesis.

If $M \equiv \lambda x.P$, then $[N/x]M \equiv M$ by 1.6(iii).

Finally, if $M \equiv \lambda y.P$, with $y \not\equiv x$, then $x \notin FV(P)$ by 1.5(iv), so again, $[N/x]M \equiv M$ by 1.6(iv)a.

(iii): If M is an atom, then $M \equiv x$ by 1.5(i),(ii), and then $FV([N/x]M) = FV(N) = FV(N) \cup (FV(M) \setminus \{x\})$ by 1.5(i). If $M \equiv PQ$, then $x \in FV(P) \cup FV(Q)$, so the induction hypothesis can be applied to at least one of P, Q, giving

$$(\star) \qquad FV([N/x]P) \cup FV([N/x]Q) = FV(N) \cup (FV(P) \setminus \{x\}) \cup (FV(Q) \setminus \{x\})$$

since, if, say $x \notin FV(Q)$, we still have $FV([N/x]Q) = FV(Q) = FV(Q) \setminus \{x\}$ by (ii). Hence

$$\begin{aligned}
FV([N/x]M) &= FV(([N/x]P)([N/x]Q)) && \text{1.6(ii)} \\
&= FV([N/x]P) \cup FV([N/x]Q) && \text{1.5(iii)} \\
&= FV(N) \cup ((FV(P) \cup FV(Q)) \setminus \{x\}) && \text{by } (\star) \\
&= FV(N) \cup (FV(PQ) \setminus \{x\}) && \text{1.5(iii)} \\
&= FV(N) \cup (FV(M) \setminus \{x\})
\end{aligned}$$

as required. Finally, if $M \equiv \lambda y.P$, then $y \not\equiv x \in FV(P)$ by 1.5(iv), and either $y \notin FV(N)$, and then

$$\begin{aligned}
FV([N/x]M) &= FV(\lambda y.[N/x]P) && \text{1.6(iv)b}\\
&= FV([N/x]P) \setminus \{y\} && \text{1.5(iv)}\\
&= (FV(N) \cup (FV(P) \setminus \{x\})) \setminus \{y\} && \text{induction hypothesis}\\
&= FV(N) \cup ((FV(P) \setminus \{x\}) \setminus \{y\}) && y \notin FV(N)\\
&= FV(N) \cup ((FV(P) \setminus \{y\}) \setminus \{x\})\\
&= FV(N) \cup (FV(M) \setminus \{x\}) && \text{1.5(iv)}
\end{aligned}$$

or $y \in FV(N)$, and then, for a fresh variable z,

$$FV([z/y]P) = \begin{cases} FV(P) & \text{if } y \notin FV(P) \\ \{z\} \cup (FV(P) \setminus \{y\}) & \text{if } y \in FV(P) \end{cases} \tag{2}$$

by (ii) or the induction hypothesis. In either case, $x \in FV([z/y]P)$, and the induction hypothesis can be applied to get

$$FV([N/x][z/y]P) = FV(N) \cup (FV([z/y]P) \setminus \{x\}). \tag{3}$$

Another immediate consequence of (2) is

$$FV([z/y]P) \setminus \{z\} = FV(P) \setminus \{y\}. \tag{4}$$

Applying these, we have

$$\begin{aligned}
FV([N/x]M) &= FV(\lambda z.[N/x][z/y]P) && \text{1.6(iv)c}\\
&= FV([N/x][z/y]P) \setminus \{z\} && \text{1.5(iv)}\\
&= (FV(N) \cup (FV([z/y]P) \setminus \{x\})) \setminus \{z\} && (3)\\
&= FV(N) \cup ((FV([z/y]P) \setminus \{x\}) \setminus \{z\}) && z \notin FV(N)\\
&= FV(N) \cup ((FV([z/y]P) \setminus \{z\}) \setminus \{x\})\\
&= FV(N) \cup ((FV(P) \setminus \{y\}) \setminus \{x\}) && (4)\\
&= FV(N) \cup (FV(M) \setminus \{x\}) && \text{1.5(iv)}
\end{aligned}$$

completing the proof of (iii).

(iv): Because of (i) and (ii), we may assume $y \not\equiv x \in FV(M)$.

If M is an atom different from x, then $[y/x]M \equiv M$, and if $M \equiv x$, then $[y/x]M \equiv y$, so $lg([y/x]M) = 1 = lg(M)$ by 1.8(i). If $M \equiv (PQ)$, then $[y/x]M \equiv ([y/x]P[y/x]Q)$ and hence

$$lg([y/x]M) = lg([y/x]P[y/x]Q) = lg([y/x]P) + lg([y/x]Q) = lg(P) + lg(Q) = lg(PQ)$$

by 1.8(ii) and the induction hypothesis. Finally, if $M \equiv \lambda z.P$, then $z \not\equiv x$, and either $y \not\equiv z$, in which case

$$lg([y/x]M) = lg(\lambda z.[y/x]P) = 1 + lg([y/x]P) = 1 + lg(P) = lg(M)$$

by 1.8(iii) and the induction hypothesis, or $M \equiv \lambda y.P$, and then

$$lg([y/x]M) = lg(\lambda w.[y/x][w/y]P) = 1 + lg([y/x][w/y]P)$$
$$= 1 + lg([w/y]P) = 1 + lg(P) = lg(M)$$

where w is the first variable not in $FV(yP)$, again by 1.8(iii) and the induction hypothesis. □

Proposition 1.10. *If x, y and z are distinct variables, and no variable that occurs in M is free in zN, then*

(i) $[N/z][z/x]M \equiv [N/x]M$ *if* $z \notin FV(M)$;
(ii) $[x/z][z/x]M \equiv M$ *if* $z \notin FV(M)$;
(iii) $[P/x][Q/y]M \equiv [([P/x]Q)/y][P/x]M$ *if* $y \notin FV(P)$;
(iv) $[P/x][Q/y]M \equiv [Q/y][P/x]M$ *if* $y \notin FV(P)$, $x \notin FV(Q)$;
(v) $[P/x][Q/x]M \equiv [([P/x]Q)/x]M$.

Proof. (i) If M is an atom, then either $M \equiv x$, and then the RHS is N and the LHS is $[N/z]z \equiv N$ by 1.6(i)a, or $M \not\equiv x$, and then the RHS is M and the LHS is $[N/z]M \equiv M$ by 1.6(i)b since $z \notin FV(M)$ and hence $z \not\equiv M$.

If M is an application PQ, then

$$[N/z][z/x](PQ) \equiv [N/z](([z/x]P)([z/x]Q))$$
$$\equiv ([N/z][z/x]P)([N/z][z/x]Q) \equiv ([N/x]P)([N/x]Q) \equiv [N/x](PQ)$$

by 1.6(ii) and the induction hypothesis.

If $M \equiv \lambda x.P$, then both sides are equal to M by 1.6(iii).

If $M \equiv \lambda y.P$ with $y \not\equiv x$, then either $x \notin FV(P)$, and then again, both sides are equal to M, this time by 1.6(iv)a, or $x \in FV(P)$, and then

$$\begin{aligned}
[N/z][z/x]M &\equiv [N/z][z/x]\lambda y.P \\
&\equiv [N/z](\lambda y.[z/x]P) && \text{1.6(iv)b, } y \notin FV(z) \\
&\equiv \lambda y.[N/z][z/x]P && \text{1.6(iv)b, } z \not\equiv y \notin FV(N) \\
&\equiv \lambda y.[N/x]P && \text{induction hypothesis} \\
&\equiv [N/x]M && \text{1.6(iv)b, } x \not\equiv y \notin FV(N)
\end{aligned}$$

(ii) By (i).
The proofs of (iii)–(v) are left as (optional) exercises. □

VARIABLE CONVENTION. From now on, we identify terms that differ only in the names of bound variables, and assume that in any context, no variable occurs both free and bound. (Technically, this means dealing with equivalence classes of terms differing only in the names of bound variables, and always picking an appropriate representative.)

1.1. β-reduction.

Definition 1.11. A *β-redex*[1] is a term of the form $(\lambda x.M)N$; its *β-contractum* is $[N/x]M$. If P has a subterm that is a β-redex and Q is the result of replacing it in P with its β-contractum,

[1] *red*ucible *ex*pression

then we say that P *β-reduces* in one step to Q and write

$$P \to_\beta Q.$$

(More formally, this means that $\to_\beta$ is the smallest binary relation on λ-terms that contains all $\langle\, \beta$-redex, contractum $\rangle$ pairs, and such that if $M \to_\beta M'$, then $MN \to_\beta M'N$, $NM \to_\beta NM'$ and $\lambda x.M \to_\beta \lambda x.M'$.)

We say that P *β-reduces* to Q and write

$$P \twoheadrightarrow_\beta Q$$

when there is a finite chain of one-step β-reductions leading from P to Q. In other words, $\twoheadrightarrow_\beta$ is the reflexive transitive closure of $\to_\beta$.

Finally, P and Q are β-convertible, $P =_\beta Q$, if there is a finite chain of one-step β-reductions and inverse one-step β-reductions leading from P to Q. That is, $=_\beta$ is the equivalence relation generated by $\to_\beta$.

Proposition 1.12. *If $M =_\beta M'$, then $MN =_\beta M'N$, $NM =_\beta NM'$ and $\lambda x.M =_\beta \lambda x.M'$.*

Proof. By induction on the number of one-step β-reductions and inverse one-step β-reductions leading from M to N. □

Examples 1.13. (i) $(\lambda x.y)N \to_\beta y$

(ii) $(\lambda xy.yx)zv \twoheadrightarrow_\beta vz$ because $(\lambda xy.yx)zv \equiv (\lambda x.(\lambda y.yx))zv \to_\beta (\lambda y.yz)v \to_\beta vz$

(iii) $(\lambda x.(\lambda y.yx)z)v \to_\beta (\lambda y.yv)z \to_\beta zv$ or $(\lambda x.(\lambda y.yx)z)v \to_\beta (\lambda x.zx)v \to_\beta zv$; either way, we have $(\lambda x.(\lambda y.yx)z)v \twoheadrightarrow_\beta zv$

(iv) $(\lambda x.xx)(\lambda x.xx) \to_\beta (\lambda x.xx)(\lambda x.xx) \to_\beta (\lambda x.xx)(\lambda x.xx) \to_\beta \ldots$

Exercise 1.14.

$$(\lambda x.x(x(yz))x)(\lambda u.uv) \to_\beta (\lambda u.uv)((\lambda u.uv)(yz))(\lambda u.uv)$$
$$\to_\beta (\lambda u.uv)(yzv)(\lambda u.uv) \to_\beta yzvv(\lambda u.uv)$$

Note an important difference between the second and third examples. In the second, at each stage of the reduction, there is at most one redex awaiting contraction, while in the third there are two ways to start contracting. Here, eventually they lead to the same result, but is this always the case? The Church-Rosser theorem below gives a very satisfactory answer to this. But before we can appreciate it, we need to decide what counts as a "result". When is a reduction "finished"? The fourth example should serve as a warning that we really do need to define these notions.

Definition 1.15. A term is in (or is a) *β-normal form* (or *β-nf*) if it contains no β-redexes. If $P \twoheadrightarrow_\beta Q$ and Q is in β-nf, then we call Q *a β-nf of P*. (In light of 1.22 below, we could write "$P =_\beta Q$" in place of "$P \twoheadrightarrow_\beta Q$" here.)

Clearly, the second and third example have β-nfs, but the fourth does not.

Theorem 1.16 (Church-Rosser)**.**
If $P \twoheadrightarrow_\beta Q$ and $P \twoheadrightarrow_\beta R$, then there is a term T such that $Q \twoheadrightarrow_\beta T$ and $R \twoheadrightarrow_\beta T$.

Corollary 1.17. *If $P \twoheadrightarrow_\beta Q$ and $P \twoheadrightarrow_\beta R$ and Q, R are in β-nf, then $Q \equiv R$ (possibly modulo renaming of bound variables).*

Proof. By the Church-Rosser theorem, there is a term T such that $Q \twoheadrightarrow_\beta T$ and $R \twoheadrightarrow_\beta T$; but since Q and R are in β-nf, T must be equal to both. □

It is important to understand what this corollary does *not* say. For example, it does not say that if a term P has a β-nf Q, then all reductions starting from P will lead to Q. Here's an example: if Ω is the term $(\lambda x.xx)(\lambda x.xx)$ of our fourth example, then $(\lambda u.v)\Omega \twoheadrightarrow_\beta v$, so it does have a β-nf, but it can also be reduced to itself indefinitely, so there is a reduction-path that does not terminate.

There is a "reduction strategy" that always results in a normal form when there is one: always reducing the leftmost redex. In other words, no redex in the argument position of a β-redex is contracted. This is called *normal order* evaluation. An other strategy with a name is *applicative order* evaluation. Here, both the operator and the operand of a β-redex are reduced until they are in normal form before the redex is contracted. This does not always find a normal form, even if there is one (see the example above), but when it does, it does in fewer steps, because it avoids multiply evaluating an argument. For example, suppose that M reduces to y in a thousand steps; then $(\lambda x.xx)M \twoheadrightarrow yy$ in 2001 steps in the normal order reduction, but only 1001 steps in the applicative order reduction.

Exercise 1.18. The *reduction graph* $G(M)$ of a term M is a directed multigraph whose vertex set is $\{\, N\colon M \twoheadrightarrow N \,\}$, with an edge corresponding to each (one-step) reduction. So, for example, $G(\mathsf{I}(\mathsf{I}x))$, where $\mathsf{I} \equiv \lambda x.x$, is $\mathsf{I}(\mathsf{I}x) \rightrightarrows \mathsf{I}x \to x$.

Draw $G(M)$ for

(i) M one of the examples above
(ii) $M \equiv \omega_3\omega_3$, where $\omega_3 \equiv \lambda x.xxx$
(iii) $M \equiv WWW$, where $W \equiv \lambda xy.xyy$
(iv) $M \equiv NN$, where $N \equiv \lambda x.(\lambda y.yy)x$.

Exercise 1.19. Show that $G(M)$ being finite neither implies nor is implied by M having a normal form.

Corollary 1.20 (Church-Rosser for β-conversion)**.** *If $P =_\beta Q$, then there is a term R with $P \twoheadrightarrow_\beta R$ and $Q \twoheadrightarrow_\beta R$.*

Proof. By induction on the number n of $\to_\beta$ and $\leftarrow_\beta$ steps leading from P to Q. The base case is trivial, so suppose that the statement holds for terms that are convertible in n steps, and let P be convertible to Q in $n+1$ steps. Then there is a term P' such that $P =_\beta P'$ in n steps, and either $Q \to_\beta P'$ or the other way round. In either case, by the induction hypothesis, there is an R' with $P \twoheadrightarrow_\beta R'$ and $P' \twoheadrightarrow_\beta R'$. So if $Q \to_\beta P'$, then R' is a good choice for R, and in the other case, since P' reduces to both R' and Q, by 1.16 there is an R with $R' \twoheadrightarrow_\beta R$ (and hence $P \twoheadrightarrow_\beta R$) and $Q \twoheadrightarrow_\beta R$. □

Corollary 1.21. *If P, Q are in β-nf, and $P =_\beta Q$, then $P \equiv Q$ (possibly modulo renaming of bound variables). In particular, $=_\beta$ is consistent in the sense that not all terms are $=_\beta$.*

Proof. By the previous corollary, P and Q reduce to a common term, which must then be $\equiv$ to both. The terms $\lambda x.x \not\equiv \lambda x$ are both in normal form. □

Corollary 1.22. *If Q is in β-nf, and $P =_\beta Q$, then $P \twoheadrightarrow_\beta Q$, that is, Q is a β-nf for P.*

Proof. They reduce to a common term, but Q only reduces to itself. □

Corollary 1.23. *If $P =_\beta Q$, then P and Q have the same β-nf or neither has one.*

Proof. If one of them, say P, has a normal form N, then $N =_\beta Q$, whence $Q \twoheadrightarrow_\beta N$ by 1.22. □

1.2. **Extensionality.** What if we wanted to treat programs as ordinary mathematical functions and identify those that always produce the same output given the same input? We could then introduce the following rule:

(ext) If $MP = NP$ for all λ-terms P, then $M = N$.

Exercise 1.24. Would the introduction of this rule make any difference? That is, are there unequal terms M, N with $MP = NP$ for all P?

Although (ext) is the direct translation of the idea of extensionality, it is technically somewhat problematic because of its infinitary nature. So we consider the following slight variant of it, which avoids this problem:

(ext') If $Mx = Nx$ for some $x \notin FV(MN)$, then $M = N$

the idea of course being that if $Mx = Nx$ with $x \notin FV(MN)$, then surely, $MP = NP$ for all P. Clearly, (ext') implies (ext).

Proposition 1.25. (ext) *implies* (ext')

Proof. Suppose $Mx = Nx$ for some $x \notin FV(MN)$; then $\lambda x.Mx = \lambda x.Nx$ by 1.12, and so $MP = (\lambda x.Mx)P = (\lambda x.Nx)P = NP$ for all terms P. □

Exercise 1.26. What purpose does the condition $x \notin FV(MN)$ serve in (ext')?

As an example of what either of these rules can prove, consider

(η) $\lambda x.Mx = M$ if $x \notin FV(M)$

This follows from (ext') because $(\lambda x.Mx)x = Mx$. A simple, but very satisfying result is that the converse holds, too. For suppose that $Mx = Nx$ with x not free in either M or N. Then

$$M = \lambda x.Mx = \lambda x.Nx = N$$

by (η) and the assumption.

Exercise 1.27. Assuming extensionality, show that $\lambda xy.xy = \lambda x.x$.

The reduction corresponding to (η) is

(η) $\lambda x.Mx \to M$ if $x \notin FV(M)$

and it can be shown that the Church-Rosser theorem with all its usual corollaries hold for this extensional version of $\twoheadrightarrow$ and $=$. But we stick to our original reduction and conversion, and will not assume (η) or any of its variants below.

1.3. **Consistency.** We have just seen that adding underivable equations (or reductions) may result in a "stronger", but still consistent (in the sense that not every two terms are equal) system. But this leaves open the question whether there are terms such that equating them results in inconsistency in this sense. Let's say that two terms M and N are incompatible, if $M = N$ implies that all terms are equal (i.e., β-convertible).

Example 1.28. $\mathsf{K} \equiv \lambda xy.x$ and $\mathsf{S} \equiv \lambda xyz.xz(yz)$ are incompatible; for suppose that $\mathsf{S} = \mathsf{K}$. Then for all terms M, we have $\mathsf{SI}M\mathsf{I} = \mathsf{KI}M\mathsf{I}$, where $\mathsf{I} = \lambda x.x$. But $\mathsf{KI}M\mathsf{I} \twoheadrightarrow \mathsf{II} \twoheadrightarrow \mathsf{I}$ and $\mathsf{SI}M\mathsf{I} \twoheadrightarrow \mathsf{II}(M\mathsf{I}) \twoheadrightarrow \mathsf{I}(M\mathsf{I}) \twoheadrightarrow M\mathsf{I}$. So we have $\mathsf{I} = M\mathsf{I}$ for all terms M. Choosing $M \equiv \lambda x.N$ we get $\mathsf{I} = N$ for every term N.

Example 1.29. xx is incompatible with xy. For, if $xx = xy$, then $MM = (\lambda xy.xx)MN = (\lambda xy.xy)MN = MN$ for all terms M, N, and taking $\mathsf{I} \equiv \lambda x.x$ for M we get $\mathsf{I} = \mathsf{II} = \mathsf{I}N = N$ for all terms N.

Exercise 1.30. $x(yz)$ is incompatible with $(xy)z$.

2. Programming in the λ-calculus

Definition 2.1 (Church numerals)**.** For a natural number n, $\bar{n} \equiv \lambda xy.x^n y$, where

$$f^n x = \begin{cases} x & \text{if } n = 0 \\ f(f^{n-1}x) & \text{if } n > 0. \end{cases}$$

With this definition, it is easy to define the successor function: since $\bar{n}yz = y^n z$, we want $\mathtt{suc}\,\bar{n}yz = y^{n+1}z = y(y^n z) = y(\bar{n}yz)$, so $\mathtt{suc} \equiv \lambda xyz.y(xyz)$ will do. Next, `plus` can be defined using the observation that one way to get $n + m$ from m is to apply `suc` n times to m; because of this, it comes in handy that the representation of n is a function that applies its first argument n times to its second argument. So $\mathtt{plus}\,\bar{n}\bar{m} \equiv \bar{n}\,\mathtt{suc}\,\bar{m}$; abstracting $\bar{n}$, $\bar{m}$ away from this we get $\mathtt{plus} \equiv \lambda xy.x\,\mathtt{suc}\,y$. Next up is `multiplication`, and its definition requires no new ideas: nm is $\mathtt{plus}\,m$ applied n times to 0, that is, $\mathtt{mult}\,\bar{n}\bar{m} = \bar{n}(\mathtt{plus}\,\bar{m})\bar{0}$, or, after abstraction, $\mathtt{mult} \equiv \lambda xy.x(\mathtt{plus}\,y)\bar{0}$.

Exercise 2.2. Define `exponentiation`.

Next, Booleans (`t` and `f`) are available in every programming language, but since we will mostly use them to choose between alternatives, we could as well define them as "choosers between alternatives", for example:

$$\mathtt{t} \equiv \lambda xy.x$$
$$\mathtt{f} \equiv \lambda xy.y$$

Exercise 2.3. Define "negation", that is, a term N such that $N\mathtt{t} \twoheadrightarrow_\beta \mathtt{f}$ and $N\mathtt{f} \twoheadrightarrow_\beta \mathtt{t}$.

Now we can define a "predicate" `isZero` that returns `t` for $\bar{0}$ and `f` for representations of other numbers:

$$\mathtt{isZero} \equiv \lambda x.x(\lambda y.\mathtt{f})\mathtt{t}$$

This works because

$$(\lambda x.x(\lambda y.\mathtt{f})\mathtt{t})\bar{0} \to_\beta \bar{0}(\lambda y.\mathtt{f})\mathtt{t} \twoheadrightarrow_\beta \mathtt{t}$$

(recall that $\bar{n}$ applies its first argument n times to its second argument) and

$$(\lambda x.x(\lambda y.\mathtt{f})\mathtt{t})\overline{n+1} \to_\beta \overline{n+1}(\lambda y.\mathtt{f})\mathtt{t} \twoheadrightarrow_\beta (\lambda y.\mathtt{f})^{n+1}\mathtt{t} \equiv (\lambda y.\mathtt{f})((\lambda y.\mathtt{f})^n\mathtt{t}) \to_\beta \mathtt{f}.$$

Unlike $\mathtt{suc}$, defining the predecessor function is a decidedly non-trivial programming assignment. Here it is:

$$\mathtt{pred} \equiv \lambda k.k[\lambda pu.u(\mathtt{suc}(p\mathtt{t}))(p\mathtt{t})](\lambda u.u\bar{0}\bar{0})\mathtt{f}$$

or, more readably,

$$\mathtt{pred} \equiv \lambda k.kHG\mathtt{f}$$

where

$$H \equiv \lambda pu.u(\mathtt{suc}(p\mathtt{t}))(p\mathtt{t}) \qquad \text{and} \qquad G \equiv \lambda u.u\bar{0}\bar{0}$$

To see that this works, we first prove

$$(*) \qquad H^nG\mathtt{t} \twoheadrightarrow_\beta \bar{n}$$

by induction on n. For $n = 0$ we have

$$H^nG\mathtt{t} \equiv G\mathtt{t} \twoheadrightarrow_\beta \mathtt{t}\bar{0}\bar{0} \twoheadrightarrow_\beta \bar{0}$$

by the definition of G and $\mathtt{t}$, and

$$\begin{aligned}
H^{n+1}G\mathtt{t} &\equiv H(H^nG)\mathtt{t} \\
&\to_\beta [\lambda u.u(\mathtt{suc}(H^nG\mathtt{t}))(H^nG\mathtt{t})]\mathtt{t} && \text{def of } H \\
&\to_\beta \mathtt{t}(\mathtt{suc}(H^nG\mathtt{t}))(H^nG\mathtt{t}) \\
&\twoheadrightarrow_\beta \mathtt{suc}(H^nG\mathtt{t}) && \text{def of } \mathtt{t} \\
&\twoheadrightarrow_\beta \mathtt{suc}\,\bar{n} && \text{IH} \\
&\twoheadrightarrow_\beta \overline{n+1}
\end{aligned}$$

completing the proof of $(*)$. But now we have

$$\begin{aligned}
\mathtt{pred}\,\bar{0} &\to_\beta \bar{0}HG\mathtt{f} && \text{def of } \mathtt{pred} \\
&\twoheadrightarrow_\beta G\mathtt{f} && \text{def of } \bar{0} \\
&\to_\beta \mathtt{f}\bar{0}\bar{0} && \text{def of } G \\
&\twoheadrightarrow_\beta \bar{0} && \text{def of } \mathtt{f}
\end{aligned}$$

and

$$\begin{aligned}
\mathtt{pred}\,\overline{n+1} &\to_\beta \overline{n+1}HG\mathtt{f} && \text{def of } \mathtt{pred} \\
&\twoheadrightarrow_\beta H^{n+1}G\mathtt{f} \\
&\equiv H(H^nG)\mathtt{f} \\
&\to_\beta [\lambda u.u(\mathtt{suc}(H^nG\mathtt{t}))(H^nG\mathtt{t})]\mathtt{f} && \text{def of } H \\
&\to_\beta \mathtt{f}(\mathtt{suc}(H^nG\mathtt{t}))(H^nG\mathtt{t}) \\
&\twoheadrightarrow_\beta H^nG\mathtt{t} && \text{def of } \mathtt{f} \\
&\twoheadrightarrow_\beta \bar{n} && \text{by } (*)
\end{aligned}$$

as desired.

2.1. **Recursion.** So far we have managed to define all our number-theoretic functions without using recursion. (The fact that the Church-numerals have iteration built in made this easy.) Now we want to see if we can use recursion, so we will try to define the factorial function the usual way, with recursion, even though it could be defined without it.

Exercise 2.4. Try to define `factorial` in roughly the same manner we defined `plus` and the rest.

Informally, the recursive definition of factorial should look something like this:

$$\texttt{fact} = \lambda n.\quad \texttt{if}\quad n = 0\quad \texttt{then}\quad 1\quad \texttt{else}\quad n * \texttt{fact}(n-1)$$

Written properly, it reads

$$(*) \qquad \texttt{fact} = \lambda n.\, \texttt{isZero}\ n\, \bar{1}\, (\texttt{mult}\, n\, (\texttt{fact}\, (\texttt{pred}\, n)))$$

This looks perfect, except that *it's not a definition*, just a requirement. If a term `fact` satisfies $(*)$, then it can rightly be called a program implementing the factorial function. But is there such a term? Can equation $(*)$ be solved for `fact`?

Well, it can, and this is a straightforward consequence of the fact that

$$(\dagger) \qquad \text{all equations of the form } FX = X \text{ can be solved}$$

or, in other words, all programs have a fixed point.

Let's first see how $(\dagger)$ implies the existence of `fact`. Indeed, $(*)$ just says that `fact` is a fixed point of the term

$$\lambda f n.\, \texttt{isZero}\ n\, \bar{1}\, (\texttt{mult}\, n\, (f\, (\texttt{pred}\, n)))$$

So if we can show $(\dagger)$, then we have a proper implementation of the factorial function. One might object that we may only have an existence proof for an implementation, but not the implementation itself. To avoid this objection, we will define a term Y that transforms every term F to a fixed point of F, so that we will have

$$F(YF) = YF$$

Defining Y is not hard, if one starts with the following idea (stolen from [Sto77, pp. 72–73]): if D (for "double") is the term $\lambda x.xx$ we have encountered before, then of course, $DD \to_\beta DD$. But this is almost what we want: for we can easily change D into a term D' such that $D'D' \to_\beta F(D'D')$: we just need to stick that F into the right place in D, to get $D' \equiv \lambda x.F(xx)$. We aren't finished yet, but we already have that

$$\forall F \exists X \;\; X =_\beta FX$$

and in fact,

$$\forall F \exists X \;\; X \to_\beta FX.$$

There's only a cosmetic change left to accomplish our goal: since we don't want to have a different fixpoint generator for each program, we need to abstract away that F from $D'D'$. So the final version is

$$Y \equiv \lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$$

and indeed, with this definition, we have

$$\begin{aligned} YF \equiv (\lambda y.(\lambda x.y(xx))(\lambda x.y(xx)))F &\to_\beta (\lambda x.F(xx))(\lambda x.F(xx)) \\ &\to_\beta F((\lambda x.F(xx))(\lambda x.F(xx))) \leftarrow_\beta F((\lambda y.(\lambda x.y(xx))(\lambda x.y(xx)))F) \equiv F(YF) \end{aligned}$$

In one respect, this last result is perhaps less satisfying than the previous one, because we don't have $YF \twoheadrightarrow_\beta F(YF)$. At least, Turing must have been bothered by this, because he defined another fixed point combinator,

$$\Theta \equiv AA, \text{ where } A \equiv \lambda xy.y(xxy)$$

and with this, we do get

$$\Theta F \equiv AAF \equiv (\lambda xy.y(xxy))AF \to_\beta (\lambda y.y(AAy))F \to_\beta F(AAF) \equiv F(\Theta F)$$

One can go wild with fixed point combinators, as the following proposition shows:

Proposition 2.5. *Every fixed point of $G \equiv \lambda yf.f(yf)$ is a fixed point combinator.*

Proof. If $M = GM$, then $MF = GMF \to_\beta (\lambda f.f(Mf))F \to_\beta F(MF)$. □

Corollary 2.6. *Any equation of the form $xy_1 \dots y_n = M$ can be solved for x; that is, there is a term X such that $Xy_1 \dots y_n = [X/x]M$.*

Proof. If Y is any fixed point combinator, then $X \equiv Y(\lambda xy_1 \dots y_n.M)$ is a solution, since

$$X = (\lambda xy_1 \dots y_n.M)X = \lambda y_1 \dots y_n.[X/x]M$$

and hence

$$Xy_1 \dots y_n = [y_n/y_n] \dots [y_1/y_1][X/x]M \equiv [X/x]M.$$

□

Exercise 2.7. Define λ-terms P, Q, such that $Px = P$ and $Qxy = yQx$, and show that they do what they are supposed to do!

3. Abstraction by application: Combinatory Logic

As in Section 1, we assume that there's a countable set of variables and constants, collectively called atoms, available.

Definition 3.1. A(n abstract) *CL-term* is a finite binary tree such that each leaf is labeled by an atom, every non-leaf vertex has exactly two children, and is labeled with `apply`.

Definition 3.2. The set of (concrete) CL-terms is the smallest set X that contains all atoms, such that $(PQ) \in X$ for all $P, Q \in X$. We call (PQ) an *application*.

As with λ-terms, we'll omit the outermost parentheses and agree that application associates to the left.

So the syntax of Combinatory Logic is that of the λ-calculus, but without λ-abstraction. And since there is no variable-binding operator, there are no bound variables either. All occurrences of variables in a term are free. Consequently, substitution can be defined the naïve way.

Definition 3.3 (Substitution)**.** $[N/x]M$ (the result of substituting N for the free occurrences of the variable x in M) is defined as follows:

(i) If M is an atom a, then:
  (a) $[N/x]M \equiv M$ if $a \not\equiv x$
  (b) $[N/x]M \equiv N$ if $a \equiv x$
(ii) If M is an application PQ, then $[N/x]M \equiv ([N/x]P)([N/x]Q)$

Examples 3.4. Suppose that two of our constants are K and S. Then

$$x\mathsf{KK} \qquad \mathsf{K}(\mathsf{SK})\mathsf{S}(x\mathsf{K}) \quad \text{and} \quad \mathsf{S}(\mathsf{S}(\mathsf{KS})(\mathsf{K}(\mathsf{SKK})))(\mathsf{S}(\mathsf{KK})(\mathsf{SKK}))$$

are all CL-terms.

Now we want to define some kind of reduction, as in Definition 1.11; but, with nothing remotely reminiscent of β-redexes, all our terms look as if they were in normal form. To jump-start the process of reduction, we'll *declare* some terms to be redexes, and tell what their contractums will be.

Definition 3.5. A *weak redex* is a term of the form $\mathsf{K}XY$ or $\mathsf{S}XYZ$; their *weak contractums* are X and $XZ(YZ)$, respectively. If P contains a weak redex and Q is the result of replacing it in P with its weak contractum, then we say that P *weakly reduces* in one step to Q and write

$$P \to_w Q.$$

(As in the case of β-redexes, this means that $\to_w$ is the smallest binary relation on CL-terms that contains all weak redex, contractum pairs, and such that if $M \to_w M'$, then $MN \to_w M'N$ and $NM \to_w NM'$.)

We say that P *weakly reduces* to Q and write

$$P \twoheadrightarrow_w Q$$

when there is a finite chain of one-step weak reductions leading from P to Q. That is, $\twoheadrightarrow_w$ is the reflexive transitive closure of $\to_w$.

Finally, P and Q are weakly convertible, $P =_w Q$, if there is a finite chain of one-step weak reductions and inverse one-step weak reductions leading from P to Q. That is, $=_w$ is the equivalence relation generated by $\to_w$.

Examples 3.6. Let $\mathsf{I} \equiv \mathsf{SKK}$, $\mathsf{B} \equiv \mathsf{S}(\mathsf{KS})\mathsf{K}$ and $\mathsf{C} \equiv \mathsf{S}(\mathsf{BBS})(\mathsf{KK})$. Then

(i) $\mathsf{I}X \twoheadrightarrow_w X$, because $\mathsf{I}X \equiv \mathsf{SKK}X \to_w \mathsf{K}X(\mathsf{K}X) \to_w X$;
(ii) $\mathsf{B}XYZ \twoheadrightarrow_w X(YZ)$, because $\mathsf{B}XYZ \equiv \mathsf{S}(\mathsf{KS})\mathsf{K}XYZ \to_w \mathsf{KS}X(\mathsf{K}X)YZ \to_w \mathsf{S}(\mathsf{K}X)YZ \to_w \mathsf{K}XZ(YZ) \to_w X(YZ)$; and
(iii) $\mathsf{C}XYZ \twoheadrightarrow_w XZY$, because $\mathsf{C}XYZ \equiv \underline{\mathsf{S}(\mathsf{BBS})(\mathsf{KK})XYZ} \to_w \mathsf{BBS}X(\underline{\mathsf{KK}X})YZ \to_w \underline{\mathsf{BBS}X}\mathsf{K}YZ \to_w \underline{\mathsf{B}(\mathsf{S}X)\mathsf{K}YZ} \to_w \underline{\mathsf{S}X(\mathsf{K}Y)Z} \to_w XZ(\underline{\mathsf{K}YZ}) \to_w XZY$.

for all terms X, Y, Z.

Exercise 3.7. (Hard.) Define a combinator (i) A such that for all terms X, $AX \twoheadrightarrow_w XX$ and (ii) D such that for all terms X and Y, $DXY \twoheadrightarrow_w YX$.

Next, we want to simulate abstraction, that is, to assign to each term M another term $[x].M$ such that

$$(*) \qquad ([x].M)N =_w [N/x]M$$

for all N. Now, if M is x, then the right-hand side is N, so $[x].x$ should be I from Example 3.6. If M is an atom different from x, then the right-hand side is M, so $[x].M \equiv \mathsf{K}M$ will do, for then $([x].M)N \equiv \mathsf{K}MN =_w M$. Finally, if $M \equiv PQ$, then the right-hand side of $(*)$ is

$$[N/x](PQ) \equiv ([N/x]P)([N/x]Q) =_w ([x].P)N(([x].Q)N) = \mathsf{S}([x].P)([x].Q)N$$

and so we can take $[x].(PQ)$ to be $\mathsf{S}([x].P)([x].Q)$. We summarize this in the following definition.

Definition 3.8. For every CL-term M and variable x, the CL-term $[x].M$ is defined as follows:

(i) If M is an atom a, then:
  (a) $[x].M \equiv \mathsf{K}M$ if $a \not\equiv x$
  (b) $[x].M \equiv \mathsf{I}$ if $a \equiv x$
(ii) If M is an application PQ, then $[x].(PQ) \equiv \mathsf{S}([x].P)([x].Q)$.

And the theorem we have just proved is

Theorem 3.9. *For all CL-terms M, N and variable x, we have $([x].M)N =_w [N/x]M$, and in fact, $([x].M)N \twoheadrightarrow_w [N/x]M$.*

Exercise 3.10. (Easy.) Armed with Theorem 3.9, do Exercise 3.7 again.

4. MODELS

Definition 4.1. A nonempty subset X of a partial order is *directed* if every two-element subset of X has an upper bound in X.

Corollary 4.2. *Every finite subset of a directed set X has an upper bound in X.*

Definition 4.3. A partial order $\langle D, \sqsubseteq \rangle$ is a *complete partial order* if

(i) there is an element $\bot$, the *bottom* of D, such that $\bot \sqsubseteq x$ for all $x \in X$; and
(ii) for every directed $X \subseteq D$, its supremum $\sqcup X \in D$ exists.

From now on, we'll suppress the ordering when this doesn't lead to confusion, and simply say that D is a **cpo**.

Examples 4.4.
(i) The non-negative real numbers extended with ∞, with the usual ordering and $\bot = 0$. It is useful as a simple source of counterexamples, so let's give it a name: we'll refer to it as $\mathcal{R}$.
(ii) $\langle \mathcal{P}\omega, \subseteq \rangle$ is a **cpo** with $\bot = \emptyset$ and $\sqcup X = \cup X$.
(iii) The set of partial functions from a set to another set; the ordering is $\subseteq$. Here $\bot = \emptyset$, the nowhere defined function, and for X directed, $\sqcup X = \cup X$.
(iv) (Streams) Finite and infinite 01-sequences, the finite ones possibly terminated by \$; $\bot$ is the empty string ε, and $x \sqsubseteq y$ iff x is a prefix of y.

Exercise 4.5. Show that in example 4.4(iv)

(i) if $\{x, y\}$ has an upper bound, then $x \sqsubseteq y \iff |x| \leq |y|$, where $|x|$ denotes the length of x
(ii) directed sets are chains (i.e., are linearly ordered)
(iii) if X is directed, and either $\{\, |x| : x \in X \,\}$ is bounded or X has an infinite element, then X has a maximal element.

Using the above, show that streams indeed form a **cpo**.

Exercise 4.6. Is $\langle \mathbb{N}, | \rangle$ a **cpo**?

Definition 4.7. An element e of a **cpo** D is *compact* if for every directed $X \subseteq D$, $e \sqsubseteq \sqcup X$ implies $(\exists x \in X) e \sqsubseteq x$.

$\perp$ is always compact, and is the only compact element in $\mathcal{R}$, since for $s > 0$, $\emptyset \neq X = (0, s)$ is directed (like all subsets of $\mathcal{R}$), with $s = \sqcup X$ but with no $x \in I$ such that $s \leq x$.

Proposition 4.8. *In $\mathcal{P}\omega$, the compact elements are the finite elements.*

Proof. Suppose first that $e \in \mathcal{P}\omega$ is finite, and let X be a directed set with $e \subseteq \cup X$. Then, by the finiteness of e, there are $x_1, \ldots, x_n \in X$ with $e \subseteq x_1 \cup \ldots \cup x_n$, and, since X is directed, there is an $x \in X$ with $x_1 \cup \ldots \cup x_n \subseteq x$.

Conversely, if x is infinite, then $X = \{\, y \subseteq x \colon y \text{ is finite} \,\}$ is clearly directed, with $x \subseteq \cup X$ but $(\forall y \in X) x \not\subseteq y$. □

Exercise 4.9. In the **cpo** of streams, the compact elements are the finite elements.

Definition 4.10. A **cpo** D is *algebraic* if for all $x \in D$,

(i) $\{\, y \sqsubseteq x \colon y \text{ is compact} \,\}$ is directed, and
(ii) $x = \sqcup \{\, y \sqsubseteq x \colon y \text{ is compact} \,\}$

Because of the second condition, $\mathcal{R}$ cannot be algebraic.

Proposition 4.11. *$\mathcal{P}\omega$ is algebraic.*

Proof. Clearly, for all $x \in \mathcal{P}\omega$, $X = \{\, y \subseteq x \colon y \text{ is finite} \,\}$ is directed, and $x = \cup X$. Hence we're done by 4.8. □

Exercise 4.12. The **cpo** of streams is algebraic.

Definition 4.13. The *Scott topology* on a **cpo** D is defined by declaring open those subsets N of D such that

(i) N is upward closed, and
(ii) if $X \subseteq D$ is directed and $\sqcup X \in N$, then $X \cap N \neq \emptyset$.

Exercise 4.14. The Scott topology is indeed a topology.

Exercise 4.15. In $\mathcal{R}$, the open sets are $\emptyset$, $[0, \infty]$ and the intervals $(x, \infty]$, with $x \geq 0$.

Exercise 4.16. What are the open sets in the stream **cpo** of Example 4.4(iv)?

Exercise 4.17. An element e of a **cpo** is compact iff $\{\, x \colon e \sqsubseteq x \,\}$ is open.

Proposition 4.18. *If D is an algebraic **cpo**, then the sets $N_e = \{\, x \colon e \sqsubseteq x \,\}$, with e compact form a base for the Scott topology on D.*

Proof. First, N_e is open by the previous exercise.

Now, for an open set N and any $e \in D$, we have that $e \in N$ implies $N_e \subseteq N$ by the definition of N_e, since N is upward closed. This proves the $\supseteq$ half of

$$(*) \qquad N = \cup \{\, N_e \colon e \in N \text{ is compact} \,\}$$

To see that $\subseteq$ holds, we need to show that for all $x \in N$, there is a compact $e \in N$ with $e \sqsubseteq x$. For $x \in N$, let

$$X = \{\, e \sqsubseteq x \colon e \text{ is compact} \,\},$$

a directed set with $x = \sqcup X$ by algebraicity; hence, since N is open, $X \cap N \neq \emptyset$, as required. □

Proposition 4.19. *In any* ***cpo****, $U_x = \{\, y \colon y \not\sqsubseteq x \,\}$ is open.*

Proof. It is clearly closed upward, and if X is directed and $\sqcup X \in U_x$, that is, $\sqcup X \not\sqsubseteq x$, then $X \cap U_x \neq \emptyset$, for otherwise we'd have that x is an upper bound of X, whence $\sqcup X \sqsubseteq x$, a contradiction. □

Proposition 4.20. *Continuous functions between* ***cpo****s are monotone.*

Proof. Suppose that $x \sqsubseteq y$ but $fx \not\sqsubseteq fy$ for f continuous. Then $fx \in U_{fy}$, so $x \in f^{-1''}U_{fy}$ and $f^{-1''}U_{fy}$ is open by the previous proposition, whence $y \in f^{-1''}U_{fy}$, that is, $fy \in U_{fy}$, a contradiction. □

Theorem 4.21. *A function $f \colon D \to D'$ between* ***cpo****s is continuous iff for all directed $X \subseteq D$, $\sqcup f''X$ exists and $f(\sqcup X) = \sqcup f''X$.*

Proof. $\Rightarrow$: Let $X \subseteq D$ be directed. By 4.20, $f(\sqcup X) \sqsupseteq f''X$ (i.e., $f(\sqcup X)$ is an upper bound of $f''X$). Now we need to show that $f''X \sqsubseteq y$ implies $f(\sqcup X) \sqsubseteq y$ for all $y \in D'$. But

$$f''X \sqsubseteq y \implies f''X \cap U_y = \emptyset \implies X \cap f^{-1''}U_y = \emptyset$$
$$\implies \sqcup X \notin f^{-1''}U_y \implies f(\sqcup X) \notin U_y \implies f(\sqcup X) \sqsubseteq y$$

since $f^{-1''}U_y$ is open by 4.19 and the continuity of f.

$\Leftarrow$: First, a function satisfying the condition on the right-hand side is monotone: for if $x \sqsubseteq y$, then $\{x, y\}$ is directed, and hence $f(y) = f(\sqcup\{x, y\}) = \sqcup f''\{x, y\} \sqsupseteq f(x)$.

It is easy to see that the image of a directed set under a monotone function is also directed. We'll make use this fact below.

Now we're ready to show that $f^{-1''}N$ is open when $N \subseteq D'$ is. $f^{-1''}N$ is upward closed, since $x \sqsubseteq y$ and $f(x) \in N$ implies $f(y) \in N$ and thus $y \in f^{-1''}N$ by monotonicity of f. Also, if $X \subseteq D$ is directed and $\sqcup X \in f^{-1''}N$, then $X \cap f^{-1''}N \neq \emptyset$, because

$$\sqcup X \in f^{-1''}N \implies \sqcup f''X = f(\sqcup X) \in N$$

by assumption, which implies $f''X \cap N \neq \emptyset$, that is, $X \cap f^{-1''}N \neq \emptyset$ since N is open and $f''X$ is directed. □

Proposition 4.22. *If D_1, D_2 are* ***cpo****s, then so is $D_1 \times D_2$, where the ordering is taken coordinate-wise, that is, $\langle\, x_1, x_2 \,\rangle \sqsubseteq \langle\, y_1, y_2 \,\rangle \iff x_1 \sqsubseteq_1 y_1 \wedge x_2 \sqsubseteq_2 y_2$.*

Proof. It is obvious that $D_1 \times D_2$ is a poset and that $\langle\, \bot_1, \bot_2 \,\rangle$ will serve as its bottom.

Now let $X \subseteq D_1 \times D_2$ be directed, and for $i = 1, 2$, let X_i be its ith projection (i.e., $X_1 = \{\, x_1 \colon \exists x_2 \langle\, x_1, x_2 \,\rangle \in X \,\}$ and similarly for X_2). Note that both projections are directed, for if x, y are in, say, X_1, then there are $x', y' \in D_2$ with $\langle\, x, x' \,\rangle$ and $\langle\, y, y' \,\rangle \in X$, and the first coordinate of their common upper bound is a common upper bound for x and y. Therefore $\sqcup X_i$ exists in D_i for $i = 1, 2$. We claim that $\langle\, \sqcup X_1, \sqcup X_2 \,\rangle$ is the supremum of X.

It is clearly an upper bound of X, since it is an upper bound of $X_1 \times X_2 \supseteq X$; and if $\langle y_1, y_2 \rangle$ is an upper bound of X, then y_i is an upper bound of X_i and hence $\sqcup X_i \sqsubseteq y_i$, so $\langle \sqcup X_1, \sqcup X_2 \rangle \sqsubseteq \langle y_1, y_2 \rangle$. □

Remark 4.23. The Scott topology on $D_1 \times D_2$ is not necessarily the product of the Scott topologies on D_1, D_2, unless D_1 and D_2 are algebraic.

Exercise 4.24. Show that if a subset of $D_1 \times D_2$ is open in the product topology, then it is open in the Scott topology.

Definition 4.25. For **cpo**s D and D', $[D \to D']$ denotes the set of continuous $D \to D'$ functions ordered by

$$f \sqsubseteq g \iff (\forall x \in D) f(x) \sqsubseteq' g(x)$$

Of course, $[D \to D']$ is a poset. But we'll soon see that it is actually a **cpo**.

Proposition 4.26. *If $F \subseteq [D \to D']$ is directed, then $f = (\lambda x \in D) \sqcup' \{ g(x) : g \in F \}$ is well-defined and continuous.*

Proof. To show that f is well-defined, we need to check that $X = \{ g(x) : g \in F \} \subseteq D'$ is directed, so that its supremum exists in D'. It is certainly non-empty (since F, being directed, is not), and if $g(x), h(x) \in X$, and $g, h \sqsubseteq i \in F$, then $g(x), h(x) \sqsubseteq i(x) \in X$.

Let $X \subseteq D$ be directed. By 4.21, we need to show that $f(\sqcup X) = \sqcup' f'' X$. Now

$$\begin{aligned} f(\sqcup X) = \sqcup' \{ g(\sqcup X) : g \in F \} &= \sqcup' \{ \sqcup' \{ g(x) : x \in X \} : g \in F \} \\ &= \sqcup' \{ g(x) : x \in X, g \in F \} = \sqcup' \{ \sqcup' \{ g(x) : g \in F \} : x \in X \} \\ &= \sqcup' \{ f(x) : x \in X \} = \sqcup' f'' X \end{aligned}$$

where the first and the penultimate equality holds by the definition of f, the second by 4.21 applied to elements of F (and the third and fourth by associativity of $\sqcup'$). □

Corollary 4.27. *For **cpo**s D and D', $[D \to D']$ is a **cpo** with $\sqcup F = (\lambda x \in D) \sqcup' \{ g(x) : g \in F \}$ for all $F \subseteq [D \to D']$.*

Proof. The function $(\lambda x \in D) \bot'$ is below every element, and if $F \subseteq [D \to D']$ is directed, then $(\lambda x \in D) \sqcup' \{ g(x) : g \in F \}$ is in $[D \to D']$ by the previous proposition, and is the supremum of F, since it's an upper bound of F, and

$$(\forall g \in F) g \sqsubseteq h \implies (\forall x \in D)(\forall g \in F) g(x) \sqsubseteq h(x) \implies (\forall x \in D) \sqcup' \{ g(x) : g \in F \} \sqsubseteq h(x),$$

that is, $(\lambda x \in D) \sqcup' \{ g(x) : g \in F \} \leq h$. □

Proposition 4.28. *A function $D_1 \times D_2 \to D'$ is continuous iff it is continuous in each of its arguments.*

Proof. $\Rightarrow$ Let $f : D_1 \times D_2 \to D'$ be continuous. We show that it is continuous in its first argument, that is, for all $x' \in D_2$, the function $g = (\lambda x \in D_1) f(x, x') : D_1 \to D'$ is continuous.

Let $X \subseteq D_1$ be directed. Then

$$g(\sqcup_1 X) = f(\sqcup_1 X, x') = f(\sqcup_\times (X \times \{ x' \})) = \sqcup' f'' (X \times \{ x' \}) = \sqcup' g'' X$$

where the second equality holds by 4.22 and the third by the continuity of f.

$\Leftarrow$ Suppose now that $X \subseteq D_1 \times D_2$ is directed, and for $i = 1, 2$, let X_i be its ith projection, as in 4.22. Then

$$f(\sqcup_\times X) = f(\sqcup_1 X_1, \sqcup_2 X_2) = \sqcup'\{\, f(x_1, \sqcup_2 X_2) \colon x_1 \in X_1 \,\}$$
$$= \sqcup'\{\, \sqcup'\{\, f(x_1, x_2) \colon x_2 \in X_2 \,\} \colon x_1 \in X_1 \,\} = \sqcup'\{\, f(x_1, x_2) \colon x_1 \in X_1, x_2 \in X_2 \,\} = \sqcup' f'' X$$

again by 4.22, the continuity of f in its first and then its second argument, and the last equality holds because of the following: first, $\sqsupseteq_\times$ is clear, since $X_1 \times X_2 \supseteq X$. For the other direction, if $x_i \in X_i$, for $i = 1, 2$, then $\langle\, x_1, x_2' \,\rangle$, $\langle\, x_1', x_2 \,\rangle \in X$ for some $x_i' \in X_i$; but, since X is directed, there is a $\langle\, y_1, y_2 \,\rangle \in X$ with $\langle\, x_1, x_2' \,\rangle$, $\langle\, x_1', x_2 \,\rangle \sqsubseteq_\times \langle\, y_1, y_2 \,\rangle$ and hence $f(x_1, x_2) \sqsubseteq_\times f(y_1, x_2) \sqsubseteq_\times f(y_1, y_2)$. Here we used the fact that f is monotone in each of its arguments by 4.20. □

Proposition 4.29 (Application is continuous)**.** *The function* $\mathrm{App} \colon [D_1 \to D_2] \times D_1 \to D_2$, *defined by* $\mathrm{App}(f, x) = f(x)$ *is continuous.*

Proof. By 4.28, we only need to check that App is continuous in each of its arguments. This is obvious for its second argument, since, for a fixed $f \in [D_1 \to D_2]$, the function $x \mapsto App(f, x)$ is f itself. Now fix $x \in D_1$, let $F \subseteq [D_1 \to D_2]$ be directed, and let g be the function $f \mapsto App(f, x)$. Then

$$g(\sqcup F) = \mathrm{App}(\sqcup F, x) = (\sqcup F)(x) = \sqcup_2\{\, f(x) \colon f \in F \,\} = \sqcup_2\{\, g(f) \colon f \in F \,\} = \sqcup_2 g'' F$$

by 4.27. □

Proposition 4.30 (Currying is continuous)**.** *Let* $f \in [D_1 \times D_2 \to D]$, *and let* $\hat{f} = (\lambda x \in D_1)(\lambda y \in D_2) f(x, y)$. *Then*

(i) $\hat{f} \in [D_1 \to [D_2 \to D]]$ *and*
(ii) $\lambda f . \hat{f} \in [[D_1 \times D_2 \to D] \to [D_1 \to [D_2 \to D]]]$

Proof. (i) For all $x \in D_1$, let $g_x = \hat{f}(x) = (\lambda y \in D_2) f(x, y)$; then $g_x \in [D_2 \to D]$, since f is continuous in its second argument by 4.28. Hence $\hat{f} = (\lambda x \in D_1) g_x \colon D_1 \to [D_2 \to D]$, and we need to show that it is continuous. So let $X \subseteq D_1$ be directed. Note first that then

$$(*) \qquad F = \{\, g_x \colon x \in X \,\} \subseteq [D_2 \to D] \quad \text{is directed}$$

because $(\lambda x \in D_1) g_x = \hat{f}$ is monotone according to 4.25, since f is continuous and hence monotone in its first argument.

Hence, denoting supremum in $[D_2 \to D]$ by $\sqcup'$, we have

$$\begin{aligned}
\hat{f}(\sqcup_1 X) &= (\lambda y \in D_2) f(\sqcup_1 X, y) && \text{def. of } \hat{f} \\
&= (\lambda y \in D_2) \sqcup \{\, f(x, y) \colon x \in X \,\} && f \text{ is continuous, 4.28} \\
&= (\lambda y \in D_2) \sqcup \{\, g_x(y) \colon x \in X \,\} && \text{def. of } g_x \\
&= \sqcup' \{\, g_x \colon x \in X \,\} && (*), \text{ 4.27} \\
&= \sqcup' \{\, \hat{f}(x) \colon x \in X \,\} && \text{def. of } g_x.
\end{aligned}$$

(ii) By (i), $\lambda f.\hat{f}$ maps $[D_1 \times D_2 \to D]$ into $[D_1 \to [D_2 \to D]]$. Now let $G \subseteq [D_1 \times D_2 \to D]$ be directed. Then

$$\begin{aligned}
(\lambda f.\hat{f})(\sqcup G) &= \widehat{\sqcup G} \\
&= (\lambda x \in D_1)(\lambda y \in D_2)((\sqcup G)(x,y)) && \text{def. of } \hat{f} \\
&= (\lambda x \in D_1)(\lambda y \in D_2) \sqcup \{ g(x,y) : g \in G \} && 4.27 \\
&= (\lambda x \in D_1) \sqcup \{ (\lambda y \in D_2) g(x,y) : g \in G \} && 4.27 \\
&= \sqcup \{ (\lambda x \in D_1)(\lambda y \in D_2) g(x,y) : g \in G \} && 4.27 \\
&= \sqcup \{ \hat{g} : g \in G \} && \text{def. of } \hat{} \\
&= \sqcup \{ (\lambda f.\hat{f})(g) : g \in G \}
\end{aligned}$$

so it is continuous. □

Proposition 4.31. *If $f_i \in [D \to D_i]$ for $i = 1, 2$, then $\langle\!\langle f_1, f_2 \rangle\!\rangle \in [D \to D_1 \times D_2]$, where $\langle\!\langle f_1, f_2 \rangle\!\rangle = \lambda x.\langle f_1(x), f_2(x) \rangle$.*

Proof. For $X \subseteq D$ directed, we have

$$\begin{aligned}
\langle\!\langle f_1, f_2 \rangle\!\rangle(\sqcup X) &= \langle f_1(\sqcup X), f_2(\sqcup X) \rangle \\
&= \langle \sqcup f_1'' X, \sqcup f_2'' X \rangle && f_i \text{ are continuous} \\
&= \sqcup \langle\!\langle f_1, f_2 \rangle\!\rangle'' X
\end{aligned}$$

where the last equality holds because of 4.22 and since the ith projection of $\langle\!\langle f_1, f_2 \rangle\!\rangle'' X$ is $f_i'' X$. □

Definition 4.32. A **cpo** D is *reflexive* if there are continuous functions $F : D \to [D \to D]$ and $G : [D \to D] \to D$ such that $F \circ G$ is the identity on $[D \to D]$.

Definition 4.33. Let D be a reflexive **cpo** via the maps F and G.

(i) For $x, y \in D$, let $x \cdot y = F(x)(y)$

(ii) Let ρ be a function mapping the variables into D. Then the interpretation of terms in D is defined recursively as follows:
- $[\![x]\!]_\rho = \rho(x)$
- $[\![MN]\!]_\rho = [\![M]\!]_\rho \cdot [\![N]\!]_\rho$
- $[\![\lambda x.M]\!]_\rho = G(\lambda d.[\![M]\!]_{\rho(x=d)})$

where $\rho(x = d)$ is the same as ρ except that it maps x to d.

For this definition to make sense, we have to show that

Proposition 4.34. *$\lambda d.[\![M]\!]_{\rho(x=d)} \in [D \to D]$, and in particular, $[\![M]\!]_\rho \in D$.*

Proof. We prove this by induction on M. If $M \equiv x$, then $\lambda d.[\![M]\!]_{\rho(x=d)} = \lambda d.d$, the identity function on D, which is clearly continuous. If $M \equiv y$, then $\lambda d.[\![M]\!]_{\rho(x=d)} = \lambda d.\rho(y)$, a constant function on D, which is continuous by 4.21.

Now suppose that $M \equiv PQ$, and set $g = \lambda d.(\llbracket P \rrbracket_{\rho(x=d)})$ and $h = \lambda d.(\llbracket Q \rrbracket_{\rho(x=d)})$, both continuous by the induction hypothesis. Then

$$\begin{aligned} \lambda\!\!\lambda d.\llbracket M \rrbracket_{\rho(x=d)} &= \lambda\!\!\lambda d.(g(d) \cdot h(d)) \\ &= \lambda\!\!\lambda d.(F(g(d))(h(d))) && \text{def of } \cdot \\ &= \lambda\!\!\lambda d.\mathrm{App}(F(g(d)), h(d)) && \text{def of App} \\ &= \mathrm{App} \circ \langle F \circ g, h \rangle \end{aligned}$$

is continuous by 4.29, 4.31, and since the composition of continuous functions are continuous. Next, if $M \equiv \lambda y.N$, then

$$\lambda\!\!\lambda d.\llbracket M \rrbracket_{\rho(x=d)} = \lambda\!\!\lambda d.\llbracket \lambda y.N \rrbracket_{\rho(x=d)} = \lambda\!\!\lambda d.G(\lambda\!\!\lambda e.\llbracket N \rrbracket_{\rho(x=d)(y=e)})$$

Setting $f(d,e) = \llbracket N \rrbracket_{\rho(x=d)(y=e)}$, by the induction hypothesis we have

$$(\forall d \in D)\lambda\!\!\lambda e.f(d,e) \in [D \to D] \quad \text{and} \quad (\forall e \in D)\lambda\!\!\lambda d.f(d,e) \in [D \to D]$$

that is, f is continuous in both of its arguments, hence $f \in [D \times D \to D]$ by 4.28. But then, by 4.30(i), $\hat{f} = \lambda\!\!\lambda d.\lambda\!\!\lambda e.f(d,e)$ is also continuous, and hence so is $G \circ \hat{f} = \lambda\!\!\lambda d.(G(\lambda\!\!\lambda e.f(d,e))) = \lambda\!\!\lambda d.(G(\lambda\!\!\lambda e.\llbracket N \rrbracket_{\rho(x=d)(y=e)}))$, as required.

Finally, if $M \equiv \lambda x.N$, then

$$\lambda\!\!\lambda d.\llbracket M \rrbracket_{\rho(x=d)} = \lambda\!\!\lambda d.\llbracket \lambda x.N \rrbracket_{\rho(x=d)} = \lambda\!\!\lambda d.G(\lambda\!\!\lambda e.\llbracket N \rrbracket_{\rho(x=d)(x=e)}) = \lambda\!\!\lambda d.G(\lambda\!\!\lambda e.\llbracket N \rrbracket_{\rho(x=e)})$$

is a constant (and hence continuous) function with value $G(\lambda\!\!\lambda e.\llbracket N \rrbracket_{\rho(x=e)})$, which is well-defined, because $\lambda\!\!\lambda e.\llbracket N \rrbracket_{\rho(x=e)} \in [D \to D]$ by the induction hypothesis. □

Exercise 4.35. If $x \notin FV(M)$, and ρ and ρ' differ at most on x, then $\llbracket M \rrbracket_\rho = \llbracket M \rrbracket_{\rho'}$.

Because of this, we may write $\llbracket M \rrbracket$ for $\llbracket M \rrbracket_\rho$ when M is a closed term.

Exercise 4.36. If M and N differ only in names of bound variables, then $\llbracket M \rrbracket_\rho = \llbracket N \rrbracket_\rho$.

Definition 4.37. Let D be a reflexive **cpo**. We say that $D \models M = N$ if $\llbracket M \rrbracket_\rho = \llbracket N \rrbracket_\rho$ for every evaluation ρ into D.

Proposition 4.38. $\llbracket M \rrbracket_{\rho(x=\llbracket N \rrbracket_\rho)} = \llbracket [N/x]M \rrbracket_\rho$

Proof. By induction on M. Throughout, let ρ' denote $\rho(x = \llbracket N \rrbracket_\rho)$.
$M \equiv x$: $LHS = \llbracket x \rrbracket_{\rho'} = \llbracket N \rrbracket_\rho = RHS$
$M \equiv y$: both sides are equal to $\rho(y)$
$M \equiv PQ$:

$$\begin{aligned} LHS = \llbracket P \rrbracket_{\rho'} \cdot \llbracket Q \rrbracket_{\rho'} &= \llbracket [N/x]P \rrbracket_\rho \cdot \llbracket [N/x]Q \rrbracket_\rho \\ &= \llbracket [N/x]P[N/x]Q \rrbracket_\rho = \llbracket [N/x]PQ \rrbracket_\rho = RHS \end{aligned}$$

by 4.33(ii), the induction hypothesis, and 1.6(ii).
$M \equiv \lambda x.P$: $LHS = \llbracket \lambda x.P \rrbracket_\rho = RHS$ by 4.35 and 1.6(iii).

$M \equiv \lambda y.P$: If $x \notin FV(P)$, then again, $LHS = [\![\lambda y.P]\!]_\rho = RHS$ by 4.35 and 1.6(iv)a. If $x \in FV(P)$, then we may assume by 4.36 that $y \notin FV(N)$, and then

$$LHS = [\![\lambda y.P]\!]_{\rho'} = G(\lambda\!\!\lambda d.[\![P]\!]_{\rho'(y=d)}) = G(\lambda\!\!\lambda d.[\![[N/x]P]\!]_{\rho(y=d)}) = [\![\lambda y.[N/x]P]\!]_\rho = RHS$$

by 4.33(ii), the induction hypothesis (applied to P and $\rho(y = d)$), and 1.6(iv)b. □

Proposition 4.39. $[\![\lambda x.M]\!]_\rho \cdot u = [\![M]\!]_{\rho(x=u)}$

Proof.

$$[\![\lambda x.M]\!]_\rho \cdot u = F(G(\lambda\!\!\lambda d.[\![M]\!]_{\rho(x=d)}))(u) = (\lambda\!\!\lambda d.[\![M]\!]_{\rho(x=d)})u = [\![M]\!]_{\rho(x=u)}.$$

□

Theorem 4.40. *$M =_\beta N$ implies $D \models M = N$.*

For the purposes of this theorem, it's useful to have the following alternative characterization of $=_\beta$:

- $(\lambda x.M)N =_\beta [N/x]M$
- if $M =_\beta N$, then $PM =_\beta PN$, $MP =_\beta NP$ and $\lambda x.M =_\beta \lambda x.N$
- $M =_\beta M$
- if $M =_\beta N$, then $N =_\beta M$
- if $M =_\beta N$ and $N =_\beta P$, then $M =_\beta P$

Proof. We show $D \models (\lambda x.M)N = [N/x]M$ and that $D \models M = N$ implies $D \models \lambda x.M = \lambda x.N$; checking the rest is routine.

$$\begin{aligned}
[\![(\lambda x.M)N]\!]_\rho &= [\![\lambda x.M]\!]_\rho \cdot [\![N]\!]_\rho \\
&= [\![M]\!]_{\rho(x=[\![N]\!]_\rho)} && \text{by 4.39} \\
&= [\![[N/x]M]\!]_\rho && \text{by 4.38}
\end{aligned}$$

If $D \models M = N$, then $D \models \lambda x.M = \lambda x.N$: Suppose that $[\![M]\!]_\rho = [\![N]\!]_\rho$ for every evaluation ρ. Then

$$[\![\lambda x.M]\!]_\rho = G(\lambda\!\!\lambda d.[\![M]\!]_{\rho(x=d)}) = G(\lambda\!\!\lambda d.[\![N]\!]_{\rho(x=d)}) = [\![\lambda x.N]\!]_\rho$$

where the middle equality holds because $[\![M]\!]_{\rho(x=d)} = [\![N]\!]_{\rho(x=d)}$ for all $d \in D$. □

Proposition 4.41. *If D is a reflexive **cpo** via F and G, then*

$$[D \to D] = \{\, F(d) : d \in D \,\},$$

that is, the continuous functions on D are exactly the representable ones.

Proof. $\subseteq$: F is onto $[D \to D]$ since $F \circ G = Id_{[D \to D]}$.
$\supseteq$: By definition, F maps into $[D \to D]$. □

Proposition 4.42. *If D is a reflexive **cpo** via F and G, then D is extensional iff $G \circ F = Id_D$. Here, extensionality means that for $d, e \in D$, $d \cdot x = e \cdot x$ for all $x \in D$ implies $d = e$.*

Proof. $\Rightarrow$: By extensionality, to see that $G(F(d)) = d$, it suffices to show that $G(F(d)) \cdot x = d \cdot x$ for every $x \in D$; but this is true, since $F(G(F(d))) = F(d)$ by 4.32.
$\Leftarrow$: Suppose that $d \cdot x = e \cdot x$ for all $x \in D$; then $F(d) = F(e)$, whence, by assumption, $d = G(F(d)) = G(F(e)) = e$. □

4.1. **The graph model.**

Proposition 4.43. *If D is an an algebraic* ***cpo****, and $f \in [D \to D]$, then*

$$f(x) = \sqcup\{\, f(e) \colon e \sqsubseteq x, e \text{ is compact} \,\}.$$

In particular, if $f \in [\mathcal{P}\omega \to \mathcal{P}\omega]$, then $f(x) = \cup\{\, f(e) \colon e \subseteq_\omega x \,\}$. Hence continuous functions on $\mathcal{P}\omega$ are determined by their values on finite sets.

Proof.

$$f(x) = f(\sqcup\{\, e \colon e \sqsubseteq x, e \text{ is compact} \,\}) = \sqcup f''\{\, e \colon e \sqsubseteq x, e \text{ is compact} \,\}$$
$$= \sqcup\{\, f(e) \colon e \sqsubseteq x, e \text{ is compact} \,\}$$

by algebraicity, and the continuity of f. □

Definition 4.44 (Coding of pairs and finite sets)**.**

(i) For $n, m \in \omega$, $(n, m) = \frac{1}{2}(n + m)(n + m + 1) + m$.
(ii) For $n \in \omega$, $e_n = \{\, i \colon k_i = 1 \,\}$ where $n = \sum_i k_i 2^i$ with $k_i \in \{\, 0, 1 \,\}$.

Exercise 4.45. (i) $\lambda nm.(n, m)$ is a bijection from ω^2 to ω
(ii) $\lambda n.e_n$ is a bijection from ω to the finite subsets of ω

Definition 4.46. (i) For $f \in [\mathcal{P}\omega \to \mathcal{P}\omega]$, $\mathrm{graph}(f) = \{\, (n, m) \colon m \in f(e_n) \,\}$.
(ii) $\mathrm{fun} = \hat{h}$, where $h(x, y) = \{\, m \colon (\exists n)(n, m) \in x, e_n \subseteq y \,\}$ for $x, y \in \mathcal{P}\omega$.

Example 4.47. $[\![\lambda x.x]\!]_\rho = \mathrm{graph}(\lambda d.[\![x]\!]_{\rho(x=d)}) = \mathrm{graph}(\lambda d.d) = \{\, (n, m) \colon m \in e_n \,\}$

Theorem 4.48. (i) $\mathrm{fun}(\mathrm{graph}(f)) = f$ *for all* $f \in [\mathcal{P}\omega \to \mathcal{P}\omega]$.
(ii) $\mathrm{fun} \in [\mathcal{P}\omega \to [\mathcal{P}\omega \to \mathcal{P}\omega]]$
(iii) $\mathrm{graph} \in [[\mathcal{P}\omega \to \mathcal{P}\omega] \to \mathcal{P}\omega]$.

In other words, $\mathcal{P}\omega$ is reflexive via fun and graph.

Proof. (i) For all $x \in \mathcal{P}\omega$,

$$\begin{aligned}
\mathrm{fun}(\mathrm{graph}(f))(x) &= \mathrm{fun}(\{\, (n, m) \colon m \in f(e_n) \,\})(x) && \text{def of graph} \\
&= \{\, m \colon (\exists n)(n, m) \in \{\, (n, m) \colon m \in f(e_n) \,\}, e_n \subseteq x \,\} && \text{def of fun} \\
&= \{\, m \colon (\exists n) m \in f(e_n), e_n \subseteq x \,\} \\
&= \cup\{\, f(e_n) \colon e_n \subseteq x \,\} \\
&= f(x) && \text{4.45(ii), 4.43}
\end{aligned}$$

(ii) By 4.30(i), it suffices to show that $h \colon \mathcal{P}\omega \times \mathcal{P}\omega \to \mathcal{P}\omega$ of Definition 4.46 is continuous; and by 4.28, that is true if h is continuous in each of its arguments.

So let's first check that for every $y \in \mathcal{P}\omega$, $g = \lambda x.h(x,y)$ is continuous. If $X \subseteq \mathcal{P}\omega$ is directed, then

$$\begin{aligned} g(\cup X) &= h(\cup X, y) \\ &= \{\, m\colon (\exists n)(n,m) \in \cup X, e_n \subseteq y \,\} && \text{def of } h \\ &= \cup_{x\in X}\{\, m\colon (\exists n)(n,m) \in x, e_n \subseteq y \,\} \\ &= \cup_{x\in X} h(x,y) && \text{def of } h \\ &= \cup_{x\in X} g(x) \\ &= \cup g''X \end{aligned}$$

Now fix $x \in \mathcal{P}\omega$, let $g = \lambda y.h(x,y)$, and let $Y \subseteq \mathcal{P}\omega$ be directed. Then

$$\begin{aligned} g(\cup Y) &= h(x, \cup Y) \\ &= \{\, m\colon (\exists n)(n,m) \in x, e_n \subseteq \cup Y \,\} && \text{def of } h \\ &= \{\, m\colon (\exists y \in Y)(\exists n)(n,m) \in x, e_n \subseteq y \,\} && Y \text{ is directed, } e_n \text{ is compact} \\ &= \cup_{y\in Y}\{\, m\colon (\exists n)(n,m) \in x, e_n \subseteq y \,\} \\ &= \cup_{y\in Y} h(x,y) && \text{def of } h \\ &= \cup_{y\in Y} g(y) \\ &= \cup g''Y \end{aligned}$$

as required.

(iii) For $F \subseteq [\mathcal{P}\omega \to \mathcal{P}\omega]$ directed, we have

$$(*) \qquad (\sqcup F)(x) = \sqcup\{\, f(x)\colon f \in F \,\} = \cup\{\, f(x)\colon f \in F \,\}$$

for all $x \in \mathcal{P}\omega$ by 4.27 and 4.4(ii); so

$$\begin{aligned} \text{graph}(\sqcup F) &= \{\, (n,m)\colon m \in (\sqcup F)(e_n) \,\} && \text{def of graph} \\ &= \{\, (n,m)\colon m \in \cup\{\, f(e_n)\colon f \in F \,\} \,\} && (*) \\ &= \cup_{f\in F}\{\, (n,m)\colon m \in f(e_n) \,\} \\ &= \cup_{f\in F}\text{graph}(f) \\ &= \cup\text{graph}''F \end{aligned}$$

□

Proposition 4.49. *The graph model is not extensional.*

Proof. Let h be as in 4.46, and let x be any subset of $\{\, (n,1)\colon n \in \omega \,\}$ containing $(0,1)$. Then, since $e_0 = \emptyset$, we have

$$x \cdot y = h(x,y) = \{\, m\colon (\exists n)(n,m) \in x, e_n \subseteq y \,\} = \{1\}$$

for all $y \in \mathcal{P}\omega$. In particular, $\{(0,1)\} \cdot y = \{(0,1),(1,1)\} \cdot y$ for all $y \in \mathcal{P}\omega$, even though $\{(0,1)\} \neq \{(0,1),(1,1)\}$ since $\lambda nm.(n,m)$ is 1-1 by 4.45(i). □

4.2. **Fixed points in the graph model.** Recall from section 2 that

$$Y \equiv \lambda y.DD, \qquad \text{where} \qquad D = \lambda x.y(xx)$$

Proposition 4.50. $[\![D]\!]_\rho = G(\lambda\!\!\lambda d.F(\rho(y))(F(d)(d)))$

Proof. Using the definition of $[\![\cdot]\!]_\rho$ (Definition 4.33), we get

$$\begin{aligned}
[\![D]\!]_\rho &= [\![\lambda x.y(xx)]\!]_\rho \\
&= G(\lambda\!\!\lambda d.[\![y(xx)]\!]_{\rho(x=d)}) \\
&= G(\lambda\!\!\lambda d.F([\![y]\!]_{\rho(x=d)})([\![xx]\!]_{\rho(x=d)})) \\
&= G(\lambda\!\!\lambda d.F(\rho(y))(F([\![x]\!]_{\rho(x=d)})([\![x]\!]_{\rho(x=d)}))) \\
&= G(\lambda\!\!\lambda d.F(\rho(y))(F(d)(d))).
\end{aligned}$$

□

Theorem 4.51. *In every reflexive* ***cpo*** *E,*

$$[\![Y]\!] \cdot u = u \cdot ([\![Y]\!] \cdot u)$$

for every $u \in E$.

Note that since F is onto $[E \to E]$, this means that $F([\![Y]\!])$ is a fixpoint operator in E; also note that this does not follow from E being a model, since there is no guarantee that every element of E is the value of a λ-term.

Proof. Let ρ be any evaluation into E. Then

$$\begin{aligned}
[\![DD]\!]_\rho &= F([\![D]\!]_\rho)([\![D]\!]_\rho) \\
&= F(G(\lambda\!\!\lambda d.F(\rho(y))(F(d)(d))))([\![D]\!]_\rho) && 4.50 \\
&= (\lambda\!\!\lambda d.F(\rho(y))(F(d)(d)))([\![D]\!]_\rho) && F \circ G = Id \\
&= F(\rho(y))(F([\![D]\!]_\rho)([\![D]\!]_\rho)),
\end{aligned}$$

that is,

(1) $$[\![DD]\!]_\rho = F(\rho(y))(F([\![D]\!]_\rho)([\![D]\!]_\rho)).$$

Also,

(2) $$[\![Y]\!]_\rho \cdot u = [\![DD]\!]_{\rho(y=u)}.$$

by 4.39. Putting these together, we get

$$\begin{aligned}
[\![Y]\!]_\rho \cdot u &= [\![DD]\!]_{\rho(y=u)} && \text{by (2)} \\
&= F(u)(F([\![D]\!]_{\rho(y=u)})([\![D]\!]_{\rho(y=u)})) && \text{by (1)} \\
&= F(u)([\![DD]\!]_{\rho(y=u)}) \\
&= F(u)([\![Y]\!]_\rho \cdot u) && \text{by (2)} \\
&= u \cdot ([\![Y]\!]_\rho \cdot u)
\end{aligned}$$

as desired. □

Proposition 4.52. $(n,m) \in e_i \implies e_i \nsubseteq e_n$

Proof. Note that

(1) $e_i \subseteq e_j \implies i \leq j$
(2) $n \in e_i \implies n < i$ (because $n < 2^i$); and
(3) $n \leq (n,m)$ (because $(n,m) \geq (n,0) = \frac{1}{2}n(n+1) \geq n$).

Now suppose that $(n,m) \in e_i$; then $n \leq (n,m) < i$ by (3) and (2), but $e_i \subseteq e_n$ would imply $i \leq n$ by (1). □

Theorem 4.53. *In $\mathcal{P}\omega$, $[\![Y]\!]$ is the "minimal fixed-point function", that is, $[\![Y]\!] \cdot u$ is the smallest fixpoint of* $\mathrm{fun}(u)$ *for every $u \in \mathcal{P}\omega$.*

Proof. By 4.51, $[\![Y]\!] \cdot u$ is a fixpoint of $\mathrm{fun}(u)$. As in the proof of 4.51, with $D = \lambda x.y(xx)$, we have

$$[\![Y]\!]_\rho \cdot u = [\![DD]\!]_{\rho(y=u)}.$$

for every evaluation ρ. Now let $d = [\![D]\!]_{\rho(y=u)}$. First, we claim that

$$d \cdot d = \cup\{\, e_i \cdot e_i : e_i \subseteq d \,\} \tag{1}$$

This is because

$$\begin{aligned}
\mathrm{fun}(d)(d) &= (\sqcup\{\, \mathrm{fun}(e) : e \subseteq_\omega d \,\})(d) && \text{4.48, 4.43}\\
&= \cup\{\, \mathrm{fun}(e)(d) : e \subseteq_\omega d \,\} && \text{4.27}\\
&= \cup\{\, \cup\{\, \mathrm{fun}(e)(e') : e' \subseteq_\omega d \,\} : e \subseteq_\omega d \,\} && \text{4.43}\\
&= \cup\{\mathrm{fun}(e)(e') : e, e' \subseteq_\omega d \,\}\\
&= \cup\{\mathrm{fun}(e)(e) : e \subseteq_\omega d \,\} && \text{monotonicity}\\
&= \cup\{\mathrm{fun}(e_i)(e_i) : e_i \subseteq_\omega d \,\} && \text{4.45}
\end{aligned}$$

Now let a be a fixpoint of $\mathrm{fun}(u)$. Because of (1), to show that $d \cdot d \subseteq a$, it suffices to show that

$$\forall i(e_i \subseteq d \implies e_i \cdot e_i \subseteq a) \tag{2}$$

We prove (2) by induction. The induction hypothesis is that

$$\forall e_n \subsetneq e_i(e_n \subseteq d \implies e_n \cdot e_n \subseteq a) \tag{2_i}$$

So let $e_i \subseteq d$ and $m \in e_i \cdot e_i$; we want to show that $m \in a$. Now, by the definition of fun, $m \in e_i \cdot e_i = \mathrm{fun}(e_i)(e_i)$ implies that

$$\exists n((n,m) \in e_i \wedge e_n \subseteq e_i)$$

by the first conjunct and 4.52, $e_i \nsubseteq e_n$, and by this and the second conjunct, we have $e_n \subsetneq e_i (\subseteq d)$. So ($2_i$) can be applied to e_n to get $e_n \cdot e_n \subseteq a$. From this, we get

$$u \cdot (e_n \cdot e_n) = \mathrm{fun}(u)(e_n \cdot e_n) \subseteq \mathrm{fun}(u)(a) = a \tag{3}$$

by the monotonicity of $\mathrm{fun}(u)$. Also, by $(n,m) \in e_i \subseteq d$ and the definition of fun,

$$m \in d \cdot e_n = u \cdot (e_n \cdot e_n)$$

where the last equation holds because

$$\begin{aligned} d \cdot e_n &= \mathrm{fun}([\![D]\!]_{\rho(y=u)})(e_n) && \text{def of } d \\ &= \mathrm{fun}(\mathrm{graph}(\lambda\!\!\lambda s.F(u)(F(s)(s))))(e_n) && 4.50 \\ &= (\lambda\!\!\lambda s.F(u)(F(s)(s)))(e_n) && \mathrm{fun} \circ \mathrm{graph} = Id \\ &= F(u)(F(e_n)(e_n)) \\ &= u \cdot (e_n \cdot e_n). \end{aligned}$$

Hence $m \in a$, and the proof is complete. □

APPENDIX: PROOF OF THE CHURCH-ROSSER THEOREM

Definition 4.54. The binary relation $\rhd$ between λ-terms is defined inductively as follows:

Rule 1. $M \rhd M$
Rule 2. if $M \rhd M'$ then $\lambda x.M \rhd \lambda x.M'$
Rule 3. if $M \rhd M'$ and $N \rhd N'$ then $MN \rhd M'N'$
Rule 4. if $M \rhd M'$ and $N \rhd N'$ then $(\lambda x.M)N \rhd [N'/x]M'$

Proposition 4.55. *If $M \rhd M'$ and $N \rhd N'$ then $[N/x]M \rhd [N'/x]M'$.*

Proof. By induction on the definition of $M \rhd M'$.

$M \rhd M'$ is a direct consequence of Rule 1. In this case we have $M' \equiv M$, so we need to prove $[N/x]M \rhd [N'/x]M$; we will do it by induction on the structure of M.

(i) If M is an atom a, then:
  (a) If $a \not\equiv x$ $[N/x]M \equiv M \rhd M \equiv [N'/x]M$ by (i)a and Rule 1.
  (b) If $a \equiv x$ then $[N/x]M \equiv N \rhd N' \equiv [N'/x]M$ by (i)b and assumption.
(ii) If M is an application PQ, then $[N/x]M \equiv ([N/x]P[N/x]Q) \rhd ([N'/x]P[N'/x]Q) \equiv [N'/x]M$ by (ii), the induction hypothesis, and Rule 3.
(iii) If M is an abstraction $\lambda x.P$, then $[N/x]M \equiv M \rhd M \equiv [N'/x]M$ by (iii) and Rule 1.
(iv) If M is an abstraction $\lambda y.P$ with $y \not\equiv x$, then:
  (a) if $x \notin FV(P)$ then $[N/x]M \equiv M \rhd M \equiv [N'/x]M$ by (iv)a and Rule 1
  (b) if $x \in FV(P)$ then $[N/x]M \equiv \lambda y.[N/x]P \rhd \lambda y.[N'/x]P \equiv [N'/x]M$ by (iv)b (since $y \notin FV(N)$ by the variable convention), the induction hypothesis, and Rule 2.

$M \rhd M'$ is a direct consequence of Rule 2. Then $M \equiv \lambda y.P$ and $M' \equiv \lambda y.P'$ where $P \rhd P'$. If $y \equiv x$, then $[N/x]M \equiv M \rhd M \equiv [N'/x]M$ by (iii) and Rule 1; otherwise, $[N/x]M \equiv \lambda y.[N/x]P \rhd \lambda y.[N'/x]P' \equiv [N'/x]M$ by (iv)b (since $y \notin FV(N)$ by the variable convention), the induction hypothesis, and Rule 2.

$M \rhd M'$ is a direct consequence of Rule 3. Then $M \equiv PQ$ and $M' \equiv P'Q'$, where $P \rhd P'$ and $Q \rhd Q'$. So $[N/x]M \equiv [N/x]P[N/x]Q \rhd [N'/x]P'[N'/x]Q' \equiv [N'/x]M$ by (ii), the induction hypothesis, and Rule 3.

$M \rhd M'$ is a direct consequence of Rule 4. Then $M \equiv (\lambda y.P)Q$ and $M' \equiv [Q'/y]P'$, where $P \rhd P'$ and $Q \rhd Q'$. If $y \equiv x$, then

$$\begin{aligned} [N/x]M &\equiv (\lambda x.P)[N/x]Q && \text{(ii) and (iii)} \\ &\rhd [([N'/x]Q')/x]P' && \text{IH, Rule 4} \\ &\equiv [N'/x][Q'/x]P' && \text{1.10(v)} \\ &\equiv [N'/x]M' \end{aligned}$$

and if $y \not\equiv x$, then either $x \notin FV(P)$, and then

$$\begin{aligned} [N/x]M &\equiv (\lambda y.P)[N/x]Q && \text{(ii) and (iv)a} \\ &\rhd [([N'/x]Q')/y]P' && \text{IH, Rule 4} \\ &\equiv [N'/x][Q'/y]P' && \text{1.10(iii)} \\ &\equiv [N'/x]M' \end{aligned}$$

or $x \in FV(P)$, and then

$$\begin{aligned} [N/x]M &\equiv [N/x](\lambda y.P)[N/x]Q && \text{(ii)} \\ &\equiv (\lambda y.[N/x]P)[N/x]Q && \text{(iv)b} \\ &\rhd (\lambda y.[N'/x]P')[N'/x]Q' && \text{IH, Rule 2,3} \\ &\rhd [([N'/x]Q')/y]P' && \text{Rule 4} \\ &\equiv [N'/x][Q'/y]P' && \text{1.10(iii)} \\ &\equiv [N'/x]M' \end{aligned}$$

□

Proposition 4.56. (i) *$\lambda x.M \rhd N$ implies $N \equiv \lambda x.M'$, where $N \rhd M'$*

(ii) *$MN \rhd L$ implies either (a) $L \equiv M'N'$, where $M \rhd M'$ and $N \rhd N'$, or (b) $M \equiv \lambda x.P$ and $L \equiv [N'/x]P'$, where $P \rhd P'$ and $N \rhd N'$.*

Proof. (i)We distinguish four cases, depending on which rule gave $\lambda x.M \rhd N$:

Rule 1. then M' is M, which is fine by Rule 1.
Rule 2. then we're done
Rule 3. can't happen, because $\lambda x.M$ is not an application
Rule 4. can't happen, for the same reason.

(ii) As before, but now with $MN \rhd L$:

Rule 1. then $L \equiv MN$, so $M' \equiv M$ and $N' \equiv N$ will do by Rule 1.
Rule 2. can't happen, because MN is not an abstraction
Rule 3. this is exactly what (a) says
Rule 4. this is exactly what (b) says.

□

Proposition 4.57. *If $M \rhd M_1$ and $M \rhd M_2$, then there is an M_3 with $M_1 \rhd M_3$ and $M_2 \rhd M_3$.*

Proof. By induction on the derivation of $M \rhd M_1$, again distinguishing cases according to the last step of its derivation.

Rule 1. In this case, $M_3 \equiv M_2$ will do.

Rule 2. Here $M \equiv \lambda x.P$ and $M_1 \equiv \lambda x.P_1$, where $P \rhd P_1$. Because of 4.56(i), $M_2 \equiv \lambda x.P_2$, and $P \rhd P_1$. Hence, by the induction hypothesis, there is a P_3 with $P_1 \rhd P_3$ and $P_2 \rhd P_3$; but then, by Rule 2, $M_1, M_2 \rhd \lambda x.P_3$.

Rule 3. Here $M \equiv PQ$, $M_1 \equiv P_1Q_1$, and $P \rhd P_1$, $Q \rhd Q_1$. By 4.56(ii), there are two subcases:

(a) $M_2 \equiv P_2Q_2$, where $P \rhd P_2$, $Q \rhd Q_2$. By the induction hypothesis, there are P_3, Q_3, with $P_1, P_2 \rhd P_3$ and $Q_1, Q_2 \rhd Q_3$, so, by Rule 3, $M_1, M_2 \rhd P_3Q_3$.

(b) $P \equiv \lambda x.R$, $M_2 \equiv [Q_2/x]R_2$, where $R \rhd R_2$, $Q \rhd Q_2$. By 4.56(i), $P_1 \equiv \lambda x.R_1$, $R \rhd R_1$. By the induction hypothesis there are Q_3, R_3 with $R_1, R_2 \rhd R_3$, $Q_1, Q_2 \rhd Q_3$; but then $M_3 \equiv [Q_3/x]R_3$ will do, because $M_1 \equiv (\lambda x.R_1)Q_1 \rhd [Q_3/x]R_3$ by Rule 4, and $[Q_2/x]R_2 \rhd [Q_3/x]R_3$ by 4.55.

Rule 4. $M \equiv (\lambda x.R)Q$ and $M_1 \equiv [Q_1/x]R_1$, where $R \rhd R_1$ and $Q \rhd Q_1$. By 4.56(ii), there are two subcases:

(a) $M_2 \equiv P_2Q_2$, where $\lambda x.R \rhd P_2$, $Q \rhd Q_2$. By 4.56(i), $P_2 \equiv \lambda x.R_2$, where $R \rhd R_2$. By the induction hypothesis, there are Q_3, R_3 with $R_1, R_2 \rhd R_3$, $Q_1, Q_2 \rhd Q_3$; but this means that $M_3 \equiv [Q_3/x]R_3$ will do, because $M_1 \rhd M_3$ by 4.55 and $M_2 \rhd M_3$ by Rule 4.

(b) $M_2 \equiv [Q_2/x]R_2$, where $R \rhd R_2$, $Q \rhd Q_2$. By the induction hypothesis there are Q_3, R_3 with $R_1, R_2 \rhd R_3$, $Q_1, Q_2 \rhd Q_3$; but then $M_3 \equiv [Q_3/x]R_3$ will do, because $M_1, M_2 \rhd M_3$ by 4.55.

□

REFERENCES

[Bar81] H. P. Barendregt. *The Lambda Calculus, its Syntax and Semantics*. North-Holland, Amsterdam, 1981.

[HS08] R. Hindley and J. Seldin. *Lambda-Calculus and Combinators, an Introduction*. Cambridge University Press, 2008.

[Sto77] Joseph E. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*. MIT Press, 1977.