The relation between Combinatory Logic and λ-calculus

In combinatory logic one can define abstraction operations that satisfy the β-scheme; such abstraction operations determine translations of the λ-calculus (\(\lambda \beta\) or \(\lambda \eta\)) into combinatory logic. Conversely the combinators correspond with λ-terms, so that CL lies embedded in \(\lambda \beta\). Because the combinators of the λ-calculus reduce stepwise, for instance, \(Kx \to \lambda y.x\) whereas \(Kx\) is not a redex, and, more importantly, because λ-abstraction is an operation of the λ-calculus, these two translations do not produce a complete agreement between CL and \(\lambda \beta\). Generally speaking, combinatory logic is weaker. Curry [CL] conceived additional axioms that close the gap: a finite number (five, to be precise) of ground equations. *(Ground means variable-free.)*

Barendregt sketches in [LC] another, more direct approach to the problem. It has inspired the present account.

Let \(P\) be a term of CL (a combinatory term), and \(M\) a λ-term. We denote by \(V(P)\) the set of variables occurring in \(P\), and by \(FV(M)\) the set of variables occurring free in \(M\). If \(V(P) = \emptyset\), we call \(P\) a ground term. We abbreviate \(SKK\) to \(I\), and note that \(Ix \to w. x\).

We use \(\approx\) for formal equality and, informally, for provable equality (in a theory determined by the context); and for convertibility, where sometimes rules will be indicated by subscripts.

The deductive systems

In all the theories considered here, there is some syntax defining terms, including application, represented by juxtaposition; and terms \(s, t\) can be combined in equations \(s \approx t\). All theories contain the schemes

\[
t \approx t \quad \text{(identity),}
\]

\[
s \approx t \Rightarrow t \approx s \quad \text{(symmetry),}
\]

\[
r \approx s \& s \approx t \Rightarrow r \approx t \quad \text{(transitivity),}
\]

\[
r \approx s \& t \approx u \Rightarrow rt \approx su \quad \text{(application);}
\]

beyond this, in \(\lambda \beta\) we have \((\lambda x.M)N \approx [N/x]M\) (β-contraction) and the \(\xi\)-rule \(M \approx N \Rightarrow \lambda x.M \approx \lambda x.N\); and additionally in \(\lambda \beta \eta\): \(\lambda x.Mx \approx M\), where \(x \not\in FV(M)\) (η-contraction). Change of bound variables in λ-terms is considered part of the syntax, and in every context (in particular that of β-contraction), the bound variables are assumed to be distinct from the free. In CL we have the combinator schemes \(KPQ \approx P\) and \(SPQR \approx PR(QR)\).

Curry presents a list of defining clauses for abstraction in combinatory logic. Four are relevant to us:

\[
\begin{align*}
(a) &\quad [x].P &= KP & \text{if } x \not\in V(P) \\
(b) &\quad [x].x &= I \\
(c) &\quad [x].Px &= P & \text{if } x \not\in V(P) \\
(f) &\quad [x].PQ &= S([x].P([x].Q))
\end{align*}
\]

Depending on which clauses we use, and the order in which we apply them, we get different definitions of abstraction. In particular, \(\lambda^* x. P\) results from (abf) — so \(\lambda^* x. P\) is \(KP\) if \(x \not\in V(P)\), \(I\) if \(P = x\), and \(S(\lambda^* x. P)(\lambda^* x. Q)\) otherwise —, \(\lambda_{1x}\) results from (bfa), and \(\lambda_{2x}\) from (abcf). Yet a fourth abstraction, \(\lambda^* x\), is de-
scribed by Hindley and Seldin [LCCI]. Call a combinatory term $P$ \textit{functional} if $Px$, $Px$ or $Px$ is a \textit{redex}. Consider the following clauses:

(c') \hspace{0.5cm} [x].Px = P \hspace{0.5cm} \text{if } x \notin V(P) \text{ and } P \text{ is functional};

(f') \hspace{0.5cm} [x].PQ = S(\lambda_{2x}.P)(\lambda_{2x}.Q).

Then $\lambda^x x$ results from (abc$'$f$'$).

Let $G_C$ be the term groupoid of combinatory logic, and $G_\lambda$ the groupoid of $\lambda$-terms. We define homomorphisms $\lambda: G_C \rightarrow G_\lambda$ en $\kappa: G_\lambda \rightarrow G_C$ as follows:

$\nu_\lambda = \nu_\kappa = \nu$, for any variable $v$;

$K_\lambda = \kappa, S_\lambda = S$;

$(\lambda.x.M)_\kappa = [x].M_\kappa$.

Actually, the precise nature of $\kappa$ depends on the details of abstraction; we might distinguish $\kappa^*, \kappa_1, \kappa_2$, and $\kappa^+$, corresponding with the choices $\lambda^x x, \lambda_1 x$, $\lambda_2 x$, and $\lambda^x x$. Observe that $P$ and $P_\lambda$, and $M$ and $M_\kappa$, contain the same variables free.

All these constructions fulfil the purpose for which they were designed:

\textbf{Lemma 1} (Reduction Lemma). Suppose

$$Q \in \{\lambda^x .P, \lambda_1 .P, \lambda_2 .P, \lambda^x .P\}.$$

Then any combinatory term $R$ satisfies $QR \rightarrow_w [R/x]P$.

The abstraction $\lambda_2$ will not lead to a system that is equivalent to $\lambda \beta$; if $y \neq x$, $\lambda_2 x . y x = y$, but the equality $\lambda x . y x \approx y$ is not valid in the $\lambda \beta$-calculus. It is \textit{almost} valid, though:

\textbf{Proposition 1}. For each $\lambda$-term $M$, $\lambda.x.(\lambda.x.M)_x \rightarrow_\beta \lambda.x.M$.

This corresponds with the fact (to be established) that the abstraction $\lambda^x x$ fits $\lambda \beta$.

\textbf{Proposition 2}. Let $P$ be a combinatory term. If $P$ is functional, then $P_\lambda \beta$-reduces to an abstraction.

\textbf{Proof}. Consider cases: $P$ is of one of the forms $K, KA, S, SA$ or $SAB$.

\textbf{Corollary}. For any combinatory term $P$, $(\lambda^x .P)_\lambda \beta$-reduces to an abstraction.

\textbf{Proof}. By definition, $\lambda^x x . P$ is functional.

For any set $A$ of ground equations, let $CL + A$ be the equational system that results from adding the axioms $A$ to $CL$.

We fix (apart from $I = SKK$) the following abbreviations:

$F := KL$, so $Fx \rightarrow I$;

$X := S(KK)$, so that $Xxy \rightarrow K(xy)$;

$Y := S(KS)$, hence $Yxy \rightarrow S(xy)$;

$U := Y(S(KY)(S(KX)S))$, hence $Ux \rightarrow S(Y(X(Sx)))$ and $Uxyz \rightarrow S(K(Sxz))(yz)$;

and $B := KY$, so that $Bxyz \rightarrow x(yz)$.
$A_0$ is the following set of axioms:

(A.1) $K = S(Y(XK))F$

(A.2) $S \simeq SU(KF)$

(A.3) $S(Y(XB))(KK) \simeq X$

(A.4) $XY \simeq X(SBF)$

(A.5) $S(KY)(YY) \simeq S(Y(X(Y(S(KY)S))))(KS)$

**Lemma 2.** $CL + (A.2) \vdash S(Y(Xx))F \simeq S(K(S(Y(Xx))F))I$.

**Proof:** $S(Y(Xx))F \simeq SU(KF)(Y(x))F$ by (A.2); now normalize.

**Lemma 3.** Suppose $P$ is functional. Then

$CL + (A.1, 2) \vdash P \simeq S(KP)I$.

**Proof:** There are five cases: $P$ is of one of the five forms $K, KA, S, SA$ or $SAB$.

(i) $K \simeq S(Y(XK))F$ by (A.1)

$\approx S(K(S(Y(XK))F))I$ by Lemma 2, with $x = K$

$\approx S(KKI) = XI$ by (A.1).

(ii) Suppose $P = KA$. Then $P \simeq S(Y(XK))FA$ by (A.1)

$\approx Y(XK)A(FA) \approx S(XKA)I \approx S(KP)I$.

(iii) $S \simeq SU(KF)U(KF)$ (use (A.2) to substitute for the initial $S$ in (A.2))

$\approx UU(KFU)(KF) \approx UUF(KF) \approx S(K(SU(KF))))(F(KF))$

$\approx S(KSI) = YI$ by (A.2).

(iv) Suppose $P = SA$. Then $P \approx SU(KF)A$ by (A.2)

$\approx UU(AKF) \approx S(Y(X(SA)))F \approx S(Y(XP))F$

$\approx S(K(S(Y(XP))F))I$ by Lemma 2

$\approx S(KP)I$ since $P \approx S(Y(XP))F$.

(v) Suppose $P = SAB$. Then $P \approx S(Y(X(SA)))FB$ by the proof of (iv)

$\approx Y(X(SA))B(FB) \approx S(X(SA)B)I \approx S(KP)I$.

Nested abstractions behave as in the $\lambda$-calculus:

**Lemma 4.** Let $P, Q$ be combinatory terms, and $x, y$ distinct variables; assume $y \notin V(P)$. Then

(i) $[P/x](\lambda_2y.Q) = \lambda_2y.[P/x]Q$;

(ii) $CL + (A.1, 2) \vdash [P/x](\lambda^2y.Q) \approx \lambda^2y.[P/x]Q$.

**Proof.** (i) By induction on $Q$, following the various cases in the definition of $\lambda_2$-abstraction.

(ii) The cases $Q = y$ and $y \notin V(Q)$ are like (i). In the third case it may be that $Q = R y$ with $y \notin V(R)$ but $R$ is not functional, whereas $[P/x]R$ is functional. Then we must show that $[P/x]S(KR)I \approx [P/x]R$; we use Lemma 3. In the final case, use (i).

**Lemma 5.** Let $P$ be a combinatory term that does not contain $x$. Then

$CL + (A.3) \vdash \lambda_1x. P \approx KP$.
Proof. Induction on the complexity of \( P \). In particular, suppose that \( P \) is composite, \( P = P_1P_2 \). Take \( y \not\in V(Px) \). Then

\[
\lambda_{1x}.P = S(\lambda_{1x}.P_1)(\lambda_{1x}.P_2) \approx S(KP)(KP) \quad \text{(ind. hyp.)}
\]

\[
\approx (\lambda_{2xy}.S(Kx)(Ky))P_1P_2 \quad \text{(Lemmas 1 and 4(i))}
\]

\[
= S(YXB)(KTK)P_1P_2 \approx XP_1P_2 \quad \text{(A.3)}
\]

\[
= (\lambda_{2xy}.K(xy))P_1P_2 \approx KP \quad \text{(Lemmas 1 and 4(i) again)}. \]

Lemma 6. Let \( P \) be a functional term that does not contain \( x \). Then

\[
CL + (A.1-3) \vdash \lambda_{1x}.Px \approx P.
\]

Proof. \( \lambda_{1x}.Px = S(\lambda_{1x}.P)I \approx S(KP)I \) by the previous lemma; by Lemma 3, \( S(KP)I \approx P \).

A major difference between \( CL \) and \( \lambda \beta \) is the \( \xi \)-rule. We want to show that it holds for \( \lambda_1 \) in \( CL + A_\beta \).

Lemma 7. (i) \( CL + (A.4) \vdash \lambda^+_y z. S(Xy)z \approx \lambda^+_y z. Xy \); (ii) \( CL + (A.5) \vdash \lambda^+_y z. w. S(S(Yy)z)w \approx \lambda^+_y z. w. S(Syw)(Sz)w \).

Proof. By applying the definition of \( \lambda^+ \)-abstraction, we get the axiom on display.

Observe that in (ii) and on the lefthand side of (i), \( \lambda^+ \) may be replaced by \( \lambda_2 \).

Theorem 1. If \( CL + A \vdash (A.3-5) \), then

\[
CL + A \vdash P \approx Q \Rightarrow CL + A \vdash \lambda_{1x}.P \approx \lambda_{1x}.Q.
\]

Proof. For ground terms \( P, Q \): from \( K \approx K \) and \( P \approx Q \), by lemma 5. For the schema \( KPQ \approx P \), let \( z \not\in V(P) \). Then

\[
\lambda_{1x}.KPQ = S(\lambda_{1x}.KP)(\lambda_{1x}.Q) = S(X\lambda_{1x}.P)(\lambda_{1x}.Q) \quad \text{by definition}
\]

\[
\approx (\lambda^+_y z. S(Xy)z)(\lambda_{1x}.P)(\lambda_{1x}.Q) \quad \text{by Lemmas 1 and 4}
\]

\[
\approx (\lambda^+_y z. x)(\lambda_{1x}.P)(\lambda_{1x}.Q) \quad \text{by Lemma 7(i)}
\]

\[
\approx \lambda^+_x. (\lambda_{1x}.P)x \quad \text{by Lemmas 1 and 4(ii)}
\]

\[
= \lambda_{1x}.P \quad \text{by definition}.
\]

For the schema \( SPQR \approx PR(QR) \), let \( z, w \not\in Var(P) \) and \( w \not\in Var(Q) \), then

\[
\lambda_{1x}.SPQR = S(\lambda_{1x}.SP)(\lambda_{1x}.R) = S(S(\lambda_{1x}.SP)(\lambda_{1x}.Q))(\lambda_{1x}.R)
\]

\[
= S(S(Y\lambda_{1x}.P)(\lambda_{1x}.Q))(\lambda_{1x}.R) \quad \text{by definition}
\]

\[
\approx (\lambda^+_y z. w. S(S(Yy)zw)(\lambda_{1x}.P)(\lambda_{1x}.Q)(\lambda_{1x}.R) \quad \text{by Lemmas 1, 4}
\]

\[
\approx \lambda^+_y z. w. S(Syw)(Szw)(\lambda_{1x}.P)(\lambda_{1x}.Q)(\lambda_{1x}.R) \quad \text{by Lemma 7(ii)}
\]

\[
\approx S(S(\lambda_{1x}.P)(\lambda_{1x}.R))(S(\lambda_{1x}.Q)(\lambda_{1x}.R)) \quad \text{by Lemmas 1, 4}
\]

\[
= \lambda_{1x}.PR(QR) \quad \text{by definition}.
\]

If \( P = P_1P_2 \) and \( Q = Q_1Q_2 \) and the last step in the deduction of \( P \approx Q \) was

\[
\frac{P_1 \approx Q_1 \quad P_2 \approx Q_2}{P \approx Q}
\]

then by induction hypothesis \( \lambda_{1x}.P_i \approx \lambda_{1x}.Q_i \) is provable \( (i = 1, 2) \); so

\[
S(\lambda_{1x}.P_1)(\lambda_{1x}.P_2) \approx S(\lambda_{1x}.Q_1)(\lambda_{1x}.Q_2),
\]
i.e. $\lambda_1 x. P \approx \lambda_1 x. Q$.

**Corollary I.** Let $\lambda$ be $\lambda^\circ$ or $\lambda^\circ$. P a combinatory term. Then

$$CL + A_\beta \vdash \lambda x. P \approx \lambda_1 x. P.$$  

**Proof.** By the Reduction Lemma, $(\lambda x. P)x \approx P$. So by the theorem,

$$\lambda_1 x. (\lambda x. P)x \approx \lambda_1 x. P.$$  

Since $\lambda x. P$ is functional, by Lemma 6 we have $\lambda_1 x. (\lambda x. P)x \approx \lambda x. P$.

**Corollary II.** Let $\lambda$ be $\lambda^\circ$ or $\lambda^\circ$. Then

$$CL + A_\beta \vdash P \approx Q \Rightarrow \lambda x. P \approx \lambda x. Q.$$  

**Lemma 8.** For any combinatory term $P$,

(i) $(\lambda_2 x. P)_\lambda \rightarrow_\beta P_\lambda$;

(ii) $(\lambda^\circ x. P)_\lambda \rightarrow_\beta P_\lambda$.

**Proof.** By induction on $P$, and using (i) for (ii).

**Lemma 9.** For any combinatory term $P$, $(\lambda^\circ x. P)_\lambda \approx_\beta \lambda x. P_\lambda$.

**Proof.** By the previous lemma, $(\lambda^\circ x. P)_\lambda \approx_\beta P_\lambda$. So by Rule (ξ),

$$\lambda x. (\lambda^\circ x. P)_\lambda \approx_\beta \lambda x. P_\lambda.$$  

Now apply Proposition 1 and the corollary to Proposition 2.

The next lemma and theorem are easiest if we take $\kappa$ to mean $\kappa^+$.

**Lemma 10.** Let $\kappa = \kappa^+$. For all $\lambda$-terms $M, N$ and for all variables $x$,

$$CL + (A.1, 2) \vdash [N_\kappa/x] M_\kappa \approx ([N/x] M)_\kappa.$$  

**Proof.** By induction on $M$. The least trivial case is abstraction. If $M = \lambda y. P$ (where by convention $y \neq x$ and $y \not\in \text{FV}(N)$), then

$$[N_\kappa/x] M_\kappa = [N_\kappa/x] (\lambda^\circ y. P_\kappa) \approx \lambda^\circ y. [N_\kappa/x] P_\kappa$$  

by Lemma 4(ii), for $y \not\in \text{V}(N_\kappa)$; so by induction hypothesis

$$[N_\kappa/x] M_\kappa \approx (\lambda^\circ y. ([N/x] P)_\kappa = ([N/x] M)_\kappa.$$  

**Theorem 2.** $\lambda \beta$ and $CL + A_\beta$ are equivalent, in the following sense: for all $\lambda$-terms $M, N$ and combinatory terms $P, Q$, and with $\kappa = \kappa^+$,

(i) $\lambda \beta \vdash M_\kappa \approx M$;

(ii) $P_\kappa \approx P$;

(iii) $\lambda \beta \vdash M \approx N \Rightarrow CL + A_\beta \vdash M_\kappa \approx N_\kappa$;

(iv) $CL + A_\beta \vdash P \approx Q \Rightarrow \lambda \beta \vdash P_\kappa \approx Q_\kappa$.

**Proof.** (i) By induction on $M$, use Lemma 9.

(ii) By induction on $P$, observe that $K_\kappa = K$ and $S_\kappa = S$.

(iii) ($\Rightarrow$) By induction on the length of the proof of $M \approx N$. Identity axioms translate to identity axioms, and instances of the application rule to instances of the application rule. For $\beta$-axioms $(\lambda x. M_1)_\kappa \approx [M_2/x] M_1$ we get

$$(\lambda x. M_1)_\kappa \approx (\lambda x. M_{1\kappa}) M_{2\kappa} \approx [M_{2\kappa}/x] M_{1\kappa} = ([M_2/x] M_1)_\kappa$$
by Lemma 10. If \( M = N \) is the conclusion of an instance of the \( \xi \)-rule, say \( M = \lambda x. M_0 \) and \( N = \lambda x. N_0 \), then by induction hypothesis

\[
CL + A_\beta \vdash M_{0\kappa} \equiv N_{0\kappa};
\]

so by Corollary II of Theorem 1 we have

\[
CL + A_\beta \vdash \lambda^x x. M_{0\kappa} \equiv \lambda^x x. N_{0\kappa},
\]

which is to say that \( M_\kappa \approx N_\kappa \) is deducible.

(iv) (\( \Rightarrow \)) By induction on the length of the proof of \( P \equiv Q \). Identity axioms translate into identity axioms, and instances of the application rule into instances of the application rule. The combinator schemes of \( CL \) correspond to the \( \beta \)-reductions \( \kappa MN \rightarrow_\beta M \) and \( \kappa MLNL \rightarrow_\beta MLNL \). The translations of the \( A_\beta \)-axioms are seen to be valid by straightforward calculation.

(iii) (\( \Leftarrow \)) If \( CL + A_\beta \vdash M_\kappa \approx N_\kappa \), then by the half of (iv) we just proved,

\[
\lambda \beta \vdash M_{\kappa \lambda} \approx N_{\kappa \lambda};
\]

so by (i), \( \lambda \beta \vdash M \equiv N \).

(iv) (\( \Leftarrow \)) If \( \lambda \beta \vdash P_\kappa \approx Q_\kappa \), then by (iii), \( CL + A_\beta \vdash P_{\lambda \kappa} \approx Q_{\lambda \kappa} \). So by (ii),

\[
CL + A_\beta \vdash P \equiv Q.
\]

Remark. By Theorem 1, Corollary I, the theorem holds for \( \kappa \in \{ \kappa^*, \kappa_1 \} \) as well, if we replace (ii) by

(ii') \( CL + A_\beta \vdash P_{\lambda \kappa} \equiv P \).

The extensional case

\( A_\eta \) is the following quartet of axioms:

(A.3) \( S(YXB)(KK) \equiv X \)

(A.4) \( YX \equiv X(SBF) \)

(A.5) \( S(KY)(YY) \equiv S(YX(Y(SKY)))((KS) \)

(A.6) \( SBF \equiv I \)

Lemma 11. \( CL + (A.6) \vdash x \equiv S(Kx)I \).

Proof. By (A.6), \( x \equiv Ix \equiv SBFx \equiv Bx(Fx) \equiv S(Kx)I \).

So (A.6) makes all terms functional, up to provable identity. Then Lemma 6 implies:

Lemma 12. Let \( P \) be a combinatory term that does not contain \( x \). Then

\[
CL + (A.6) \vdash \lambda_1 x. Px \equiv P.
\]

Lemma 13. For any combinatory term \( P \), \( CL + A_\eta \vdash \lambda_2 x. P \equiv \lambda_1 x. P \).

Proof. By Theorem 1, the \( \xi \)-rule holds for \( \lambda_1 \) in \( CL + A_\eta \). So from

\[
(\lambda_2 x. P)x \equiv P
\]

(\( \text{Reduction Lemma} \))

we get \( \lambda_1 x. (\lambda_2 x. P)x \equiv \lambda_1 x. P \); and by Lemma 12, \( \lambda_1 x. (\lambda_2 x. P)x \equiv \lambda_2 x. P \).

Combining Theorem 1 with Lemma 13, we obtain:

Lemma 14. \( CL + A_\eta \vdash P \equiv Q \Rightarrow CL + A_\eta \vdash \lambda_2 x. P \equiv \lambda_2 x. Q \).
Lemma 15. For any combinatory term \( P \), \((\lambda x.P)_\eta \approx_{[\eta]} \lambda x.P_\eta \).

Proof. By Lemma 8(i), \((\lambda x.P)_\eta \) \( \rightarrow_\beta P_\eta \). Then by the \( \xi \)-rule,
\[ \lambda x.(\lambda x.P)_\eta \approx_{[\eta]} \lambda x.P_\eta. \]
By the (\( \eta \))-scheme, \( \lambda x.(\lambda x.P)_\eta \approx (\lambda x.P)_\eta \).

Lemma 16. Let \( \kappa = \kappa_2 \). For all \( \lambda \)-terms \( M, N \) and any variable \( x \),
\[ [N_\kappa/x]M_\kappa = ([N/x]M)_\kappa. \]

Proof. As Lemma 10; use (i) of Lemma 4 instead of (ii).

Theorem 3. \( \lambda \beta \eta \) and \( CL + A_\eta \) are equivalent, in the following sense: for all \( \lambda \)-terms \( M, N \) and combinatory terms \( P, Q \), and with \( \kappa = \kappa_2 \),
\[ (i) \lambda \beta \eta \vdash M_\kappa \approx M; \]
\[ (ii) P_\lambda = P; \]
\[ (iii) \lambda \beta \eta \vdash M \approx N \iff CL + A_\eta \vdash M_\kappa \approx N_\kappa; \]
\[ (iv) CL + A_\eta \vdash P \approx Q \iff \lambda \beta \eta \vdash P_\lambda \approx Q_\lambda. \]

Proof. Similar to the proof of Theorem 2; use Lemmas 5 and 17-19.

Remark. The argument of Lemma 13 applies to \( \lambda^* \) or \( \lambda^+ \) as well. Hence the theorem holds for \( \kappa \in \{ \kappa^*, \kappa_1, \kappa^+ \} \) as well; only for \( \kappa^* \) and \( \kappa_1 \) we must replace (ii) by
\[ (ii') \quad CL + A_\eta \vdash P_\lambda \approx P. \]

Curry’s axioms

Curry does not use the axioms (A.1, 2); instead he has
\[ (C.1) \quad K = B(B(SBF)K), \]
\[ (C.2) \quad S \approx B(B(SBF))S. \]
These do the same job as (A.1, 2). Abbreviate \( SBF \) to \( I_\eta \).

Lemma 17. \( CL + (C.2) \vdash Bxy \approx B(Bxy)I. \)

Proof: \( Bxy \approx S(Kx)y \approx B(BI_\eta)S(Kx)y \) by (C.2)
\[ \approx BI_\eta(S(Kx)y) \approx I_\eta(S(Kx)y) \approx B(S(Kx)y)(F(S(Kx)y)) \]
\[ \approx B(Bxy)I. \]

Lemma 18. Suppose \( P \) is functional. Then
\[ CL + (C.1, 2) \vdash P \approx S(KP)I. \]

Proof: We have five cases as in Lemma 3. Observe that \( S(KP)I \approx BPI \).
\[ (i) K \approx BI_\eta K \text{ by (C.1)} \]
\[ \approx B(BI_\eta K)I \text{ by Lemma 17, with } x = I_\eta \text{ and } y = K \]
\[ \approx BKI. \]
\[ (ii) \text{ Suppose } P = KA. \text{ Then } P \approx BI_\eta KA \text{ by (C.1)} \]
\[ \approx I_\eta P \approx BPI. \]
\[ (iii) S \approx B(BI_\eta)S \text{ by (C.2)} \]
COMBINATORY LOGIC AND LAMBDA-CALCULUS

\[ \approx B(B(BI_\eta)S)I \quad \text{by Lemma 17} \]
\[ \approx BSI. \]

(iv) Suppose \( P = SA \). Then \( P \approx B(BI_\eta)SA \) \( \text{by (C.2)} \)
\[ \approx BI_\eta P \approx B(BI_\eta P)I \quad \text{by lemma 17} \]
\[ \approx BPI. \]

(v) Suppose \( P = SAB \). Then \( P \approx B(BI_\eta)SAB \) \( \text{by (C.2)} \)
\[ \approx BI_\eta (SA)B \approx I_\eta P \approx BPI. \]

References

Further References
D.A. Turner: Another algorithm for bracket abstraction. Afdruk van JSL XLIV, 267-70. A4(2r) Λ.

Gebruikt combinator \( S' \) gedefinieerd door \( S'xyzw = x(yw)(zw) \).