Combinatory logic and applicative functors

Samuel Dean

A **language for combinatory logic** is a language for predicate logic consisting of the following:

- Any number of constant symbols which we call **primitives**. We use capital letters to denote primitives.

- A binary function symbol called **application**. For terms x and y, we write xy for application of (x, y).

Application has **left fixity**, which is to say that xyz is shorthand for $(xy)z$, and $x_1x_2 \dots x_n$ is shorthand for $(x_1x_2 \dots x_{n-1})x_n$.

Every primitive P comes with a **reduction rule**, which is a sentence

$$\forall x_1, x_2, \dots, x_n, Px_1x_2 \dots x_n = t(x_1, x_2, \dots, x_n)$$

where $t(x_1, x_2, \dots, x_n)$ is a term in the variables $x_1, x_2, \dots, x_n$.

Given a language for combinatory logic $\mathcal{L}$ and a theory T which contains a reduction rule for each primitive, we call a model $\mathfrak{M} \models T$ a **combinatory algebra (without types)**. We call the elements of a combinatory algebra **combinators**.

The SK basis We write **SK** for the theory with two primitives S and K, with the reduction rules

$$\forall x, y, z, Sxyz = xz(yz)$$
$$\forall x, y, Kxy = x.$$

The theory **SK** known as the **SK basis**. Smullyman refers to K as the **kestrel** in [2].

Remark 1. We view combinators as 'functions' which act on combinators and return combinators. The reduction rule of a primitive is seen as that combinator's 'definition' as a function. In [1], Schönfinkel referred to S as the **melting function**.

We now give examples of closed terms in **SK**, and their properties. Some of them have bird names, given to them by Smullyman in CITE.

- The **idenity** $I = SKK$ satisfies $\mathbf{SK} \models \forall x, Ix = x$.

- Since combinatory application is left associative, one of the most useful combinators is the one that gives rise to right associative composition. It is very tempting sometimes to compute gfx as $g(fx)$, so we make a combinator which allows us to do so. For this, we introduce Schönfinkel's $B = S(KS)$, also known as the **bluebird**, which satisfies

$$\mathbf{SK} \models \forall x, y, z, Bxyz = x(yz).$$

- Consider the sequence of terms V_n given by

$$V_0 = I$$
$$V_{n+1} = S(KS)(S(KK)V_n).$$

One can show by induction that, for $n > 0$,

$$\mathbf{SK} \models \forall x, y, z_0, \dots, z_{n-1}, V_n x y z_0 \dots z_{n-1} = x(y z_0 \dots z_{n-1}).$$

as required.

- For any $n \in \mathbb{N}$ and $i \in n$, we define a term P_i^n by induction, by

$$\begin{aligned} &P_0^0 = I \\ &P_i^n = V_{n-1} K P_0^{n-1} \text{ if } n > 0 \\ &P_i^n = K P_{i-1}^{n-1} \text{ if } n > 0 \text{ and } i > 0. \end{aligned}$$

One can prove by induction that

$$\mathbf{SK} \models \forall P_i^n x_0 \dots x_{n-1} = x_i.$$

It is clear that the primitives of **SK** have a good deal of expressive power. It turns out that, when considered as 'functions' in the sense of Remark 1, the closed terms in **SK** can capture any term in **SK**. The proof of this reveals that S and K are not at all arbitrary: They are designed to enable currying.

Proposition 2 (Currying in **SK**)**.** *Let $t(x_0, \dots, x_{n-1})$ be any term in* **SK** *for $n > 0$. There is a term $\tau(x_0, \dots, x_{n-2})$ in* **SK** *such that*

$$\mathbf{SK} \models \forall x_0, \dots, x_{n-1}, \tau(x_0 \dots x_{n-2}) x_{n-1} = t(x_0, \dots, x_{n-1}).$$

We say that such a term τ ***curries*** *t* ***(with respect to*** *x_{n-1}**)**.*

Proof. If x_{n-1} does not occur in t, then $\tau = Kt$ curries t. Therefore, we may assume that x_{n-1} occurs in t.

If $t = ab$ for terms a and b, and α and β curry a and b respectively, then $S\alpha\beta$ curries t. Note, however, that x_{n-1} does not occur in a and $b = x_{n-1}$, then a gives a more simple currying of t. Therefore, we may assume that t is an atomic term.

We are left with $t = x_{n-1}$, which is curried by I. This completes the proof. □

Corollary 3 (Expressive power of **SK**)**.** *Let $t(x_0, \dots, x_{n-1})$ be any term in* **SK***. There is a closed term τ in* **SK** *such that*

$$\mathbf{SK} \models \forall x_0, \dots, x_{n-1}, \tau x_0 \dots x_{n-1} = t(x_0, \dots, x_{n-1}).$$

Proof. We obtain τ from t by repeated currying. □

Remark 4. The proofs of Proposition 2 and Corollary 3 provide an algorithm for turning any term in **SK** into a closed term with the same expressive power. Consider the term $t(x,y,z) = xzy$. By following this algorithm to the letter, and expanding out I into SKK gives,

$$\tau = S(S(KS)(S(KK)S))(KK)$$

However, this representation is not unique. If

$$C = S(S(K(S(KS)K))S)(KK)$$

then

$$\mathbf{SK} \models \forall x,y,z, Cxyz = xzy.$$

Note also that

$$\mathbf{SK} \models \forall x,y,z, CIxy = yx.$$

The mockingbird The **mockingbird** is the closed term

$$M = SII.$$

One can easily show that

$$\mathbf{SK} \models \forall f, Mf = ff.$$

Note, however, that it is impossible to simplify MM. If we apply the definition we end up in a loop: $MM = MM = MM = \ldots$.

The Y combinator The **Y combinator** is the closed term

$$Y = BM(CBM).$$

One can easily show that

$$\mathbf{SK} \models \forall f, f(Yf) = Yf.$$

For this reason, M is sometimes known as the **fixed point combinator**. The Y combinator has a problem more extreme than the mockingbird's: We are unable to simplify Yf for any combinator f.

Example 5 (Connection to applicative functors)**.** Let $F = \mathrm{Hom}(X, -) : \mathrm{Set} \to \mathrm{Set}$ for a set X. Then F is an applicative functor[1] with

$$(x * y)z = x(z)(y(z))$$
$$(\text{pure } x)y = x.$$

We recognise that these expressions are little more than the definitions of S and K, our fundamental combinators, written in a different notation.

References

[1] Schönfinkel, M. Über die bausteine der mathematischen logik. *Mathematische Annalen 24* (1924).

[2] Smullyan, R. *To Mock a Mockingbird.* Knopf, 1985.

[1] `https://en.wikipedia.org/wiki/Applicative\_functor\#Examples`