5 Combinatory algebras

To give a model of the lambda calculus means to provide a mathematical space in which the axioms of lambda calculus are satisfied. This usually means that the elements of the space can be understood as functions, and that certain functions can be understood as elements.

Naïvely, one might try to construct a model of lambda calculus by finding a set X such that X is in bijective correspondence with the set X^X of *all* functions from X to X. This, however, is impossible: for cardinality reason, the equation $X \cong X^X$ has no solutions except for a one-element set $X = 1$. To see this, first note that the empty set $\emptyset$ is not a solution. Also, suppose X is a solution with $|X| \geqslant 2$. Then $|X^X| \geqslant |2^X|$, but by Cantor's argument, $|2^X| > |X|$, hence X^X is of greater cardinality than X, contradicting $X \cong X^X$.

There are two main strategies for constructing models of the lambda calculus, and both involve a restriction on the class of functions to make it smaller. The first approach, which will be discussed in this section, uses *algebra*, and the essential idea is to replace the set X^X of all function by a smaller, and suitably defined set of *polynomials*. The second approach is to equip the set X with additional structure (such as topology, ordered structure, etc), and to replace X^X by a set of structure-preserving functions (for example, continuous functions, monotone functions, etc).

5.1 Applicative structures

Definition. An *applicative structure* $(\mathbf{A}, \cdot)$ is a set $\mathbf{A}$ together with a binary operation "$\cdot$".

Note that there are no further assumptions; in particular, we do *not* assume that application is an associative operation. We write ab for $a \cdot b$, and as in the lambda calculus, we follow the convention of left associativity, i.e., we write abc for $(ab)c$.

Definition. Let $(\mathbf{A}, \cdot)$ be an applicative structure. A *polynomial* with coefficients in $\mathbf{A}$ and over a set of variables $x_1, \ldots, x_n$ is a formal expression built from variables and elements of A by means of the application operation. In other words, the set of polynomials is given by the following grammar:

$$t, s \quad ::= \quad x \mid a \mid ts,$$

where x ranges over variables and a ranges over the elements of $\mathbf{A}$. We write $\mathbf{A}\{x_1, \ldots, x_n\}$ for the set of polynomials with coefficients in $\mathbf{A}$ over $x_1, \ldots, x_n$.

Here are some examples of polynomials over the variables x, y, z, where $a, b \in \mathbf{A}$:

$$x, \qquad xy, \qquad axx, \qquad (x(y(zb)))(ax).$$

If $t(x_1, \ldots, x_n)$ is a polynomial over the indicated variables, and $b_1, \ldots, b_n$ are elements of $\mathbf{A}$, then we can evaluate the polynomial at the given elements: the evaluation $t(b_1, \ldots, b_n)$ the element of $\mathbf{A}$ obtained by "plugging" $x_i = b_i$ into the polynomial, for $i = 1, \ldots, n$, and evaluating the resulting expression in $\mathbf{A}$. Note that in this way, every polynomial t in n variables can be understood as a *function* from $\mathbf{A}^n \to \mathbf{A}$. This is very similar to the usual polynomials in algebra, which can also either be understood as formal expressions or as functions.

If $t(x_1, \ldots, x_n)$ and $s(x_1, \ldots, x_n)$ are two polynomials with coefficients in $\mathbf{A}$, we say that the equation $t(x_1, \ldots, x_n) = s(x_1, \ldots, x_n)$ *holds* in $\mathbf{A}$ if for all $b_1, \ldots, b_n \in \mathbf{A}$, $t(b_1, \ldots, b_n) = s(b_1, \ldots, b_n)$.

5.2 Combinatory completeness

Definition (Combinatory completeness)**.** An applicative structure $(\mathbf{A}, \cdot)$ is *combinatorially complete* if for every polynomial $t(x_1, \ldots, x_n)$ of $n \geqslant 0$ variables, there exists some element $a \in \mathbf{A}$ such that

$$ax_1 \ldots x_n = t(x_1, \ldots, x_n)$$

holds in $\mathbf{A}$.

In other words, combinatory completeness means that every polynomial *function* $t(x_1, \ldots, x_n)$ can be represented (in curried form) by some *element* of $\mathbf{A}$. We are therefore setting up a correspondence between functions and elements as discussed in the introduction of this section.

Note that we do not require the element a to be unique in the definition of combinatory completeness. This means that we are dealing with an intensional view of functions, where a given function might have several different names in general (but see the discussion of extensionality later in this section).

The following theorem characterizes combinatory completeness in terms of a much simpler algebraic condition.

Theorem 5.1. *An applicative structure $(\mathbf{A}, \cdot)$ is combinatorially complete if and only if there exist two elements $s, k \in \mathbf{A}$, such that the following equations are satisfied for all $x, y, z \in \mathbf{A}$:*

$$\begin{array}{ll} (1) & sxyz = (xz)(yz) \\ (2) & kxy = x \end{array}$$

Example 5.2. Before we prove this theorem, let us look at a few examples.

(a) The identity function. Can we find an element $i \in \mathbf{A}$ such that $ix = x$ for all x? Yes, indeed, we can let $i = skk$. We check that for all x, $skkx = (kx)(kx) = x$.

(b) The boolean "false". Can we find an element $\mathbf{F}$ such that for all x, y, $\mathbf{F}xy = x$? Yes, this is easy: $\mathbf{F} = k$.

(c) The boolean "true". Can we find $\mathbf{T}$ such that $\mathbf{T}xy = y$? Yes, what we need is $\mathbf{T}x = i$. Therefore a solution is $\mathbf{T} = ki$. And indeed, for all y, we have $kixy = iy = y$.

(d) Find a function f such that $fx = xx$ for all x. Solution: let $f = sii$. Then $siix = (ix)(ix) = xx$.

Proof of Theorem 5.1: The "only if" direction is trivial. If $\mathbf{A}$ is combinatorially complete, then consider the polynomial $t(x, y, z) = (xz)(yz)$. By combinatory completeness, there exists some $s \in \mathbf{A}$ with $sxyz = t(x, y, z)$, and similarly for k.

We therefore have to prove the "if" direction. Recall that $\mathbf{A}\{x_1, \ldots, x_n\}$ is the set of polynomials with variables $x_1, \ldots, x_n$. For each polynomial $t \in \mathbf{A}\{x, y_1, \ldots, y_n\}$ in $n+1$ variables, we will define a new polynomial $\lambda^* x.t \in \mathbf{A}\{y_1, \ldots, y_n\}$ in n variables, as follows by recursion on t:

$$\begin{array}{lcll} \lambda^* x.x & := & i, & \\ \lambda^* x.y_i & := & ky_i & \text{where } y_i \neq x \text{ is a variable,} \\ \lambda^* x.a & := & ka & \text{where } a \in \mathbf{A}, \\ \lambda^* x.pq & := & s(\lambda^* x.p)(\lambda^* x.q). & \end{array}$$

We claim that for all t, the equation $(\lambda^* x.t)x = t$ holds in $\mathbf{A}$. Indeed, this is easily proved by induction on t, using the definition of λ^*:

$$\begin{array}{lcl} (\lambda^* x.x)x & = & ix = x, \\ (\lambda^* x.y_i)x & = & ky_i x = y_i, \\ (\lambda^* x.a)x & = & kax = a, \\ (\lambda^* x.pq)x & = & s(\lambda^* x.p)(\lambda^* x.q)x = ((\lambda^* x.p)x)((\lambda^* x.q)x) = pq. \end{array}$$

Note that the last case uses the induction hypothesis for p and q.

Finally, to prove the theorem, assume that $\mathbf{A}$ has elements s, k satisfying equations (1) and (2), and consider a polynomial $t \in \mathbf{A}\{x_1, \ldots, x_n\}$. We must show that there exists $a \in \mathbf{A}$ such that $ax_1 \ldots x_n = t$ holds in $\mathbf{A}$. We let

$$a = \lambda^* x_1. \ldots .\lambda^* x_n.t.$$

Note that a is a polynomial in 0 variables, which we may consider as an element of $\mathbf{A}$. Then from the previous claim, it follows that

$$\begin{array}{lcl} ax_1 \ldots x_n & = & (\lambda^* x_1.\lambda^* x_2. \ldots .\lambda^* x_n.t)x_1 x_2 \ldots x_n \\ & = & (\lambda^* x_2. \ldots .\lambda^* x_n.t)x_2 \ldots x_n \\ & = & \ldots \\ & = & (\lambda^* x_n.t)x_n \\ & = & t \end{array}$$

holds in $\mathbf{A}$. □

5.3 Combinatory algebras

By Theorem 5.1, combinatory completeness is equivalent to the existence of the s and k operators. We enshrine this in the following definition:

Definition (Combinatory algebra)**.** A *combinatory algebra* $(\mathbf{A}, \cdot, s, k)$ is an applicative structure $(\mathbf{A}, \cdot)$ together with elements $s, k \in \mathbf{A}$, satisfying the following two axioms:

$$\begin{array}{ll} (1) & sxyz = (xz)(yz) \\ (2) & kxy = x \end{array}$$

Remark. The operation λ^*, defined in the proof of Theorem 5.1, is defined on the polynomials of any combinatory algebra. It is called the *derived lambda abstractor*, and it satisfies the law of β-equivalence, i.e., $(\lambda^* x.t)b = t[b/x]$, for all $b \in \mathbf{A}$.

Finding actual examples of combinatory algebras is not so easy. Here are some examples:

Example 5.3. The one-element set $\mathbf{A} = \{*\}$, with $* \cdot * = *$, $s = *$, and $k = *$, is a combinatory algebra. It is called the *trivial* combinatory algebra.

Example 5.4. Recall that Λ is the set of lambda terms. Let $\mathbf{A} = \Lambda/{=_\beta}$, the set of lambda terms modulo β-equivalence. Define $M \cdot N = MN$, $S = \lambda xyz.(xz)(yz)$,