Lambda Calculus

killingform

July 25, 2018

These notes (unfinished, obviously) were of a course I took on lambda calculus, proof, and category theory.

Contents

I Introduction

1. What is a function?

There are two main roots of the theory of functions:

(a) analytic geometry (Descartes).

(b) the development of calculus (Leibniz, 1673, in a letter to Bernoulli).

Euler and Dirichlet concerned themselves more with numerical functions, and in the 19th century, Bolzano, Cauchy, Weierstrass, and Fourier all had the understanding of an analytic representation of functions:

$$\varphi(x) = \dots,$$

for some formula. It was Fourier who first broke it! The modern definition is due to set theory, namely Dedekind and Cantor. Bourbaki's definition is that a function from $A \to B$ is a triple (A, R, B) such that:

(i) A and B are sets.

(ii) $R \subseteq A \times B$ (the "graph").

(iii) $\forall x \in A \exists! y \in B \,.\, (x, y) \in R$.

Intuitively, "$R(x, y)$" $\equiv$ "$f(x) = y$". We now say that $(A, R, B) = (A', B', R')$ if and only if $A = A'$, $B = B'$, and $R = R' \subseteq A \times B$. So the modern functions are equivalence classes of such triples:

$$f = g \iff \forall x \in A \,.\, f(x) = g(x).$$

Consider the expressions:

$$2 + 3 = 5, \qquad 5 = 5.$$

The left is dynamic, however the right is tautological.

In λ-calculus, we remove A, B from the triple. So if we wrote f as:

$$f \colon A \to B \colon x \mapsto f(x),$$

λ-calculus only concerns itself with the rule of transformation, $x \mapsto f(x)$. In the way, we write f now as:

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

the λ indicating that x is the scope of the function, and that we just replace x in the rule $f(x)$ with whatever we apply $\lambda x.f(x)$ to. A few explicit examples:

$$\lambda x.x^2, \quad \lambda x.2x, \quad \lambda x.\sin x.$$

To apply a function f in Λ-calculus to something a, we write f‘a. This convention is due to Bertrand Russell. A few more examples:

$$(\lambda x.x^2)\text{‘}a = x^2\,[a/x] = a^2, \qquad (\lambda x.2x)\text{‘}b = 2x\,[b/x] = 2b,$$

where the notation $x^2\,[a/x]$ says "substitute a for x in x^2".

Since we are merely just doing substitution, we have for anything that x‘y or x‘x even makes sense! This is the untyped λ-calculus.

II Untyped λ-calculus

1. Terms and variables

We define the untyped λ-calculus by structural induction, defining the terms. The class of variables, $x_!, x_2, x_4, \ldots$, is denoted by Var (we assume it is countable). We define the **terms** as follows:

1. Every variable is a term (this is the base case of our structural induction).
2. If M, N are terms, so is $M`N$.
3. If M is a term, $x \in \text{Var}$, then $\lambda x.M$ is a term.
4. Nothing else is a term.

In this way, we have defined everything we consider a "term" (a function). In BNF (Backus-Naur Form), this is written:

$$MN ::= x \mid M`N \mid \lambda x.M$$

If you like sets a lot (I do), you can think of $\lambda x.\varphi(x)$ as the set $\{x \mid \varphi(x)\}$. In this way, we view $x`y$ as meaning $x \in y$.

Denote by $FV(M)$ the set of **free variables** of a term M, defined by induction:

1. If $x \in \text{Var}$, then $FV(x) := \{x\}$.
2. $FV(M`N) = FV(M) \cup FV(N)$.
3. $FV(\lambda x.M) = FV(M) \setminus \{x\}$.

Similarly, we can define the **bound variables** inductively as follows:

1. If $x \in \text{Var}$, then $BV(x) := \varnothing$.
2. $BV(M`N) - BV(M) \cup BV(N)$.
3. $BV(\lambda x.M) = BV(M) \cup \{x\}$.

We say that a term M is **closed** if and only if $FV(M) = \varnothing$. Another name for closed terms is **combinators**. Some examples of combinators:

$$K = \lambda x.\lambda y.x, \quad S = \lambda x.\lambda y.\lambda z.(x`z)`(y`z), \quad I = \lambda x.x.$$

We should probably discuss some notational kinks and quirks that we will use frequently. Firstly, we associate function application (`) to the left, that is:

$$((a`b)`c = a`b`c,$$

and more so, we might even drop the indicator `, so it further becomes:

$$(a`b)`c = a`b`c = abc.$$

With repeated bindings $\lambda x.$, we act from the inside out, sometimes dropping the consecutive lambdas to save space and eliminate (or sometimes create) confusion:

$$\lambda x.(\lambda y.(\lambda z.M)) = \lambda xyz.M.$$

So for example, our combinators from before with these conventions appear as:

$$K = \lambda xy.x, \quad S = \lambda xyz.(xz)(yz), \quad I = \lambda x.x.$$

An example, we can 'reduce' the term SKK as follows:

$$\begin{aligned} SKKx &= ((SK)K)x \\ &= (Kx)(Kx) \\ &= x \\ &= Ix, \end{aligned}$$

where we see that if we had the **axiom of extensionality**:

$$\frac{\forall x\,.\,(fx = gx)}{f = g},$$

then we could conclude that $SKK = I$. Otherwise, the most we can say is that $SKKx = Ix$.

2. Change of bound variables and α-congruence

The idea of α-congruence is that:

$$\lambda x.t =_\alpha \lambda y.t[y/x],$$

where y is some new/fresh variable. That is, the bound variable can be changed, and we say that the functions are the same still—this is α-congruence. A priori, we can't say that these are equal without considering α-congruence as the bound variables are different. They are not constructively the same, but we would like to say they are the same function at the end of the day.

Define $t[M/x]$ for $x \in \text{Var}$ and terms t, M, inductively by, $y \in \text{Var}$:

1. $y[M/x] = \begin{cases} M, & x = y; \\ y, & \text{otherwise.} \end{cases}$
2. $(M`N)[y/x] = (M[y/x])`(N[y/x])$.
3. $(\lambda x.M)[y/x] = \begin{cases} \lambda y.M[y/x], & x = z; \\ \lambda z.M[y/x], & x \neq z. \end{cases}$

If we apply α-congruence to λ-calculus, we get that terms are equivalent up to change of bound variables.

3. Equality, $=$, and β-equivalence

The phenomenon of α-congruence brings to mind that we need to rigorously define what it means to be equal.

Equality ($=$) is an equivalence relation, so we have the usual reflexivity, symmetry, and transitivity:

$$M = M, \quad \frac{M = N}{N = M}, \quad \frac{M = N, N = P}{M = P}.$$

In addition, we have the following:

$$\frac{M = N}{M`R = N`R}, \quad \frac{M = N}{R`M = R`N}, \quad \frac{M = N}{\lambda x.M = \lambda x.N} \quad (\xi),$$

where the last one there is the ξ-rule. In addition, we also have the α-congruence rule, discussed above:

$$\lambda x.M = \lambda y.M[y/x], \quad (\alpha)$$

and what is known as β-reduction:

$$(\lambda x.M)`N = M[N/x], \quad (\beta)$$

which says that equality respects application. Finally, we have the η-rule:

$$\lambda x.(fx) = f, \text{ for } x \notin FV(f). \quad (\eta)$$

Proposition 1. *The η-rule is equivalent to extensionality.*

Proof. Suppose η and note:

$$f`x = g`x \underset{(\xi)}{\Longrightarrow} \lambda x.(f`x) = \lambda x.(g`x) \underset{(\eta)}{\Longrightarrow} f = g.$$

Conversely, suppose EXT, and note:

$$(\lambda x.(f`x))`x = f`x, \quad (\beta)$$

for some $x \notin FV(f)$, and so by EXT: $f = \lambda x.(f`x)$. ■

A few examples of β-reduction:

$$(\lambda x.xx)`a =_\beta aa, \quad (\lambda x.xx)`(\lambda x.xx) =_\beta (\lambda x.xx)`(\lambda x.xx),$$

and as my professor said after showing us this, "We're no longer in Kansas, Toto".

Theorem 1 (Fixed point theorem)**.** *For every F there exists an X such that $F`X = X$.*

Proof. Let $W := \lambda x.F(xx)$ and $X := WW$. Note:

$$X = WW = (\lambda x.F(xx))W =_\beta F(WW) = F(X).$$

■

This gets us close to Russell's paradox, indeed only a little bit extra is needed. Consider the λ-term:

$$R := \lambda x.\mathrm{NOT}(xx),$$

where "NOT" is just the negation function (we will see this at a later date). Then we try β-reducing RR:

$$RR = (\lambda x.\mathrm{NOT}(xx))R = \mathrm{NOT}(RR).$$

Clearly not in Kansas.

More generally, define the Y-combinator:

$$Y := \lambda f.(\lambda x.f(xx))(\lambda x.f(xx)),$$

so we see that:

$$\begin{aligned} Yf &= (\lambda x.f(xx))(\lambda x.f(xx)) \\ &= f(\lambda x.f(xx))(\lambda x.f(xx)) \\ &= f(Yf). \end{aligned}$$

If we try to "type" this (by assigning domains and codomains to our function), we get:

$$Y\colon A^A \to A\colon f \mapsto Y(f).$$

Historical Remark 1. It was Church who first used the notation $\lambda x.f(x)$, and he said this was because they initially they used $\hat{x}.f(x)$ which then changed slightly to be $\wedge x.f(x)$, and then finally $\lambda x.f(x)$ in order to accommodate printers.

Another story about the early years of λ-calculus is that Church and Kleene sought to find functions for any computable functions. The issue they came across was that they couldn't figure out how to λ-tize the predecessor function:

$$n \mapsto n - 1.$$

Church had nearly almost convinced himself that it was not possible, but then his student Kleene took a trip to the dentist. There, he had a Eureka moment, and he figured out the trick to defining the predecessor function in λ-calculus!

Apart from silly stories, we can recant a little about the history of λ-calculus. Both λ-calculus and combinatory logic (logic studied via combinators) sprung to life in the 1920s. Moses Schönfinkel was probably the first to study combinatory logic. A Russian logician, he gave a talk to Hilbert and the gang on 7 December 1920 about combinatory logic. Heinrich Behmann revised and published Schönfinkel's notes in 1924 (also, Jean van Heijenoort's "From Frege to Gödel" has it). He sadly only had one other paper published, and then he returned to Moscow. In 1927, he was reportedly mentally ill and in the confines of a sanatorium. He died poor in Moscow in 1942, all his papers burned by his neighbours for heating.

Schönfinkel wrote that:

$$A \times B \to C \quad \equiv \quad A \to C^B,$$

something which we will see later is a feature of a Cartesian closed category!

Interestingly, Schönfinkel was a pupil of Samuel Shatunovsky, a mathematician who had correspondence with Brouwer on cosntructivist ideas. Even more interestingly, Shatunovsky did not have a formal school certificate so he never really had a regular research career in mathematics.

Curry dabbled in combinatory logic during 1922, and von Neumann's thesis on set theory had vague combinator-like structures in it (he also had close correspondence with the logic group at Göttingen who were likely to pass on Schönfinkel's wisdom). Curry's invention of combinatory logic likely was entirely a re-invention, independent to Europe, as Curry was in the USA.

Curry focused his attention on substitution, which at the time was not a well-analysed feature of logic. Indeed, Russell and Whitehead just forgot to define it in Principia Mathematica, and Hilbert and Ackermann's textbook had an incorrect statement for it concerning predicate logic. Curry studied in particular how to break substitution up into smaller steps, currying. This lead him to combinatory logic.

It was only in 1928 that Alonzo Church invented the λ-calculus. Church's untyped λ-calculus appeared in print in 1932 with unrestricted quantification, no law of the excluded middle. However, it was found quite quickly after that it had a contradiction. He fixed it, and at the time hoped that Gödel's incompleteness theorems for Russell and Whitehead's system did not extend to this system (see: *History of Lambda-calculus and Combinatory Logic*, Cardone & Hindley).

4. Combinatory calculus

Recall that a combinator is a closed λ-term. We have the following examples of combinators:

$$S = \lambda xyz.(xz)(yz), \quad K = \lambda xy.x, \quad I = \lambda x.x, \quad B = \lambda xyz.x(yz), \quad C = \lambda xyz.xzy.$$

We have already seen that $SK = I$ (assuming extensionality), and indeed B and C can also be obtained from S and K alone.

Exercise 1. Find, using only S and K, an expression equivalent to B and C.

It turns out that any lambda term can be expressed in terms of the combinators S and K, and hence to any computable function (Church thesis). There exists algorithms for determining the SK-expression, however we will not cover that as it is fun finding your own ways.

Historical Remark 2. These days, combinatory logic is increasingly popular with the rise of functional programming where it is actively applied.

Note that B can be seen as composition of functions if you alter the definition a little bit:

$$Bfg := f \circ g, \qquad f \circ g = \lambda x.f(gx).$$

Exercise 2. Show that The set of λ-terms with application and identity I form a monoid. Do you need η? This might actually be for function composition using the B combinator, not using application. Need to try.

One can make combinators into a form of logic by viewing K as "true", and then $K^* := \lambda xy.y$ as false. You are encouraged to find all your favourite boolean operators.

Note that both Church and Curry hated η and extensionality since it has no computational meaning.

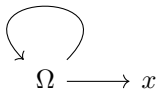
Exercise 3. Let M be a λ-term. Associate M to a directed multigraph, possibly with loops, called $\mathcal{G}(M)$, where:

(i) The vertices is the set $\{N \mid M \ \beta\text{-reduces to } N\}$.

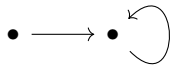
(ii) The edges are each possible β-reduction.

Example 1. If $\Omega := (\lambda x.xx)(\lambda x.xx)$, then $\mathcal{G}(\Omega)$ is:

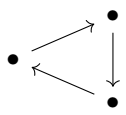
while $\mathcal{G}(Kx\Omega)$ is:

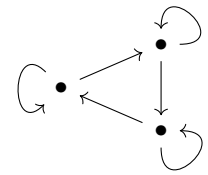


Find λ-terms M so that $\mathcal{G}(M)$ is of the following shapes:

(a) 

(b) 

(c) 

(d) 

An equational theory is said to be **inconsistent** if and only if every equation is derivable (that is, every model is a singleton).

Proposition 2. *λ-calculus with $K = S$ is inconsistent.*

Proof. Suppose that $K = S$, so $Kabc = Sabc$ for all a, b, c. Hence we have $ac = (ac)(bc)$, so in particular, $II = (II)(bI)$, which reduces to $I = bI$ for all b. Let $b = Kf$, and then we have $I = KfI \iff I = f$ for any term f. If we choose a different $b = Kg$, we similarly get $I = g$, and hence $f = g$ for any two terms. ■

Remark that there are other possible equations one might add to guarantee inconsistency. Church gives one, forming a theory where we quotient out by terms without a normal form (with respect to β-reduction). If we define:

$$T := \{M = N \mid M, N \text{ } \lambda\text{-terms without normal form}\},$$

then T is inconsistent. A hint to show this is to let $M = \lambda x.((xK)\Omega)$ and $N = \lambda x.((xS)\Omega)$. If $M = N$, show that $S = K$!