1 Introduction

The λ-calculus (here we first introduce pure untyped λ-calculus) is a formal system designed to capture of the notions of "function" and "composition".

Remark. A λ-expression will be a finite string made from the symbols λ, ., and an infinite list of variable symbols $x, y, z, \cdots$ or $a, b, c, \cdots$ or $x_1, x_2, x_3, \ldots$ whatever you want to call them. We think of $\lambda x.M$ as the function that takes in x and returns M where M is an expression possibly involving x.

Definition 1.0.1. A *well-formed* λ-expression, or λ-*term*, is defined recursively via,

(a) any variable is a λ-term

(b) if M is a λ-term then $\lambda x.M$ is a λ-term

(c) if M and N are λ-terms then (MN) is a λ-term.

The set of λ-terms is denoted Λ.

Remark. We have already said that we should interpret $\lambda x.M$ as a function. Then (M, N) is an "application" of the function M to N. We think of $((\lambda x.M)N)$ "evaluating" to $M[x := N]$ which is how this system captures the essence of functions and computation as evaluation though substitution. Although intutively we think of λ-terms as functions, that actually take an input and produce a well-defined output, this is actually difficult to define because we will have to decide when a computation is "finished" and in fact computations may not halt complicating our desire to call these things functions. A major goal will be to somehow interpret these objects as honest-to-god functions. For now, we take a different perspective not that λ-terms are "machines" but rather they are formal strings in a formal system. To create a formal system we need "rules of inference" which are conventionally called conversions and reductions.

Definition 1.0.2. A λ-term of the form $(\lambda x.M)\, N$ is called a β-*redex* and its β-*reduction* is the corresponding term $M[x := N]$. We write,

$$(\lambda x.M)\, N \rhd_{\beta,1} M[x := N]$$

If a term M can be converted to N via a finite sequence of β-reductions we write,

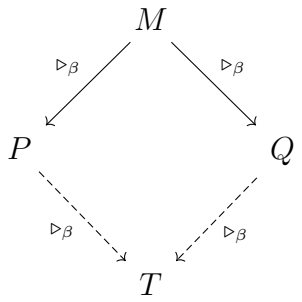
$$M \rhd_\beta N$$

Then the equivalence relation generated by this (using zigzags) is called β-equivalence and writen as $M =_\beta N$.

Example 1.0.3. $(\lambda x.x)y \rhd_{\beta,1} y$ so we say that $\lambda x.x$ is the identity function.

Definition 1.0.4. A λ-term M is a *normal form* if it does not contain any β-redices. We say that N *is a normal form* of M if N is a normal form and $M \rhd_\beta N$.

Theorem 1.0.5 (Church-Rosser)**.** Let M, P, Q be λ-terms such that $M \rhd_\beta P$ and $M \rhd_\beta Q$ then there exists a λ-term T such that $P \rhd_\beta T$ and $Q \rhd_\beta T$. We express this with the diagram,



Corollary 1.0.6. Suppose M, N are λ-terms with $M =_\beta N$ then there exists a λ-term T such that $M \triangleright_\beta T$ and $N \triangleright_\beta T$. We say that M and N "compute the same value".

Corollary 1.0.7. The normal form of M (if it exists) is unique.

Example 1.0.8. A λ-term need not admit a normal form. For example let,

$$\omega \equiv \lambda x.(xx)$$

this is the function that applies its input to itself. What happens if we apply ω to itsef then we get,

$$\Omega \equiv (\omega\omega)$$

and it is easy to see that,

$$\Omega \triangleright_{\beta,1} \Omega$$

and there are no other possible β-reductions. Therefore Ω does not have a β-normal form. We can think of it as corresponding to a computation that does not halt.

2 Arithmetic and Logic in the λ-Calculus

Functions have a built-in way of expressing natural numbers, namely counting the number of iterates of a function. Therefore we can define the Church numerals as follows,

$$\underline{n} \equiv \lambda f.\lambda x.(f^n\ x)$$

where f^n means the expression $(f\ (f\ (f\ \cdots)))$ iterated n times. We think of n as expressing the function which takes in a function f and returns its n^{th} iterate. We can use this to define sucessor,

$$\text{succ} \equiv \lambda n.(\lambda f.\lambda x.f\ (n\ f\ x))$$

and addition,

$$\text{plus} \equiv \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x) \equiv \lambda m.\lambda n.\lambda f.(m\ \text{succ})\ n$$

which is just composition of the two iterations or equivalently $(m\ \text{succ})$ which is the function that applies the sucessor function m times applied to n. However, it is multiplication which shows the true flexibility of the system. We define,

$$\text{mult} \equiv \lambda m.\lambda n.\lambda f.\lambda x.(m\ (n\ f)\ x) \equiv \lambda m.\lambda n.(m\ (\text{plus}\ n))$$

which applies m the "apply its input m times" function to $(n\ f)$ which is the function f^n so we get mn iterates of f. Equivalently we can iterate $(\text{plus}\ n)$ the "add n" function m times. Likewise,

boolean values are naturally represented by their corresponding branching behavior, true should take in two functions and return the first while false should take in two functions and do the latter such that applying a boolean executes an if-then statment. Thus,

$$\text{true} \equiv \lambda p.\lambda q.p \quad \text{and} \quad \text{false} \equiv \lambda p.\lambda q.q$$

I will leave it as an exercise to produce λ-terms expressing the basic logical operations.

Definition 2.0.1. We say that a function $f : \mathbb{N} \to \mathbb{N}$ is *represented* by an untyped λ-term F if for all $n \in \mathbb{N}$,

$$(F\ \underline{n}) =_\beta [\underline{f(n)}]$$

If there exists such an F we say that f is representable. If F can be chosen such that it has a normal form then we say that f is strongly representable.

Remark. We will see that every computable function is representable but not necessarily strongly representable.

3 "Consistency" of the λ-Calculus (SKIP)

EARLY FORM INCONSISTENT

Look at section here https://plato.stanford.edu/entries/lambda-calculus/
and in Barendregt's book.
THIS FORM IS CONSISTENT IN THE SENSE OF CANNOT DERIVE EVERYTHING
CURY'S PARADOX

4 Recursion and Fixed Points

We've discussed how λ-terms can represent integer functions. However, so far we've only implemented very simple function such as addition and multiplication. It is not at all clear how to implement complex behavior without loops. In fact, we want to be able to implement general recursion but this seems impossible without self-referential definitions which are not allowed in the construction of our formal language.

Although there does not appear to be a way to *construct* recursive functions we do have a way to check if a given λ-term represents the required recursive function using a fixed-point expression. Indeed, suppose that we want to find a function f which satisfies the recursive definiton,

$$f(x) = \begin{cases} g(f(x-1), x) & x > 0 \\ n & x = 0 \end{cases}$$

Suppose moreover that g is represented by a λ-term G. Then consider the λ-term,

$$T :\equiv \lambda f.\lambda x.(\text{IsZero } x)\ \underline{n}\ (G\ (f\ (\text{pred } x))\ x))$$

which takes a function f and returns the function expressed by the RHS of the recursive definition. Therefore, we are searching for a λ-term F such that,

$$(T\ F) =_\beta F$$

This will then represent the function f. Therefore, we have reduced the problem to constructing fixed-points of T up to $\equiv_{\beta\eta}$.

Consider the following λ-term,

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

called Haskell Curry's paradoxical combinator. Since that is quite a mouthful and its usual name has been coopted, I will simply call it "Y". Notice that, like Ω, we have,

$$(Y\ f) \rhd_{\beta,1} (f\ (Y\ f))$$

In particular, $(Y\ T)$ is a fixed point of T. Therefore, astonishingly, in untyped λ-calculus, every λ-term admits a fixed point. Hence the recursion problems allways admit solutions.

PHILOSOPHICAL MUSINGS ON THE NATURE OF THESE FIXED POINT PROBLEMS ANOTHER WAY TO GET FIXED POINTS, ARE THESE THE SAME??

To even make sense of this question, we first need a notion of convergence of functions suitable for these integer functions. Moreover, because we want to ask if the limit of the representing λ-terms agree, we need some sort of *model* in which the λ-terms are *all* representing functions which we can reason about topologically. This is the subject we now turn to.

5 λ-Models (DO AT THE END)

From the begining, the question arrose: if λ-terms are supposed to be "anonomous functions" what space are they functions on? More precisely, we want a notion of a space such that it's automorphisms are exactly the equivalence classes of λ-terms. This space must have some curious properties. Since λ-terms can be applied to any other λ-term it must be the case that each point of the space represents a function on it and every "nice" function arises in this fasion. For example, this couldn't be done in the category of sets since it would be saying there is a set X with $2^X = X$ which is impossible by Cantor's theorem. We need a more interesting notion to capture what's going on in the untyped λ-calculus.

Definition 5.0.1. A λ-*interpretation* is a set D and a *valuation* function $\rho : \mathrm{Var} \to D$

Definition 5.0.2. A λ-*model* is a triple $(D, \bullet, [\![-]\!])$ with $\bullet$ a binary operation on D and $[\![-]\!]$ a function giving for each valuation ρ a mapping $[\![-]\!]_\rho : \Lambda \to D$ which satisfies,

(a) if x is a variable $[\![x]\!]_\rho = \rho(x)$

(b) for any λ-terms M, N then $[\![M\,N]\!]_\rho = [\![M]\!]_\rho \bullet [\![N]\!]_\rho$

(c) for all variables x, terms M, and elements $d \in D$ we have $[\![\lambda x.M]\!]_\rho \bullet d = [\![M]\!]_{\rho[x:=d]}$

(d) for all terms M and valuations ρ, σ we have $[\![M]\!]_\rho = [\![M]\!]_\sigma$ if $\rho(x) = \sigma(x)$ for all free varaibles x of M

(e) for all terms M and variables x, y then $[\![\lambda x.M]\!]_\rho = [\![\lambda y.M[x := x]]\!]_\rho$ if y is not free in M

(f) for all terms M, N if for all $d \in D$ we have $[\![M]\!]_{\rho[x:=d]} = [\![N]\!]_{\rho[x:=d]}$ then $[\![\lambda x.M]\!]_\rho = [\![\lambda x.N]\!]_\rho$.

Remark. The *term model* or *trivial model* is given by Λ modulo β-equivalence and composition given by application. This is a model but it is unenlightening. Dana Scott searched for a more interesting model.

5.1 Syntax-Free Models

DO THIS

6 Domain Theory

Remark. We use "monotone" and "order-preserving" synonymously to mean $x \leq y \implies f(x) \leq f(y)$.

Definition 6.0.1. A *directed-complete partial order* (dcpo) is a poset $(P, \leq)$ such that every directed subset has a supremum.

Remark. A directed subset $D \subset P$ is a subset such that every finite subset of D has an upper bound in D. Equivalently for each $a, b \in D$ there is $c \in D$ with $a, b \leq c$.

Remark. A somewhat technical axiom of choice argument shows that it suffices to check that every chain has a supremum in order for a poset to be a dcpo.

Remark. It is good to contrast a dcpo with the related notion of the complete lattice, a poset such that every subset has a supremum. Taking the supremum over the set of lower bounds gives all infima as well. Then a complete lattice is a lattice because the supremum and infimum of $\{a, b\}$ give the meets and joins. However, $\{a, b\}$ is *not* directed and therefore we should not expect a dcpo to be either a meet or a join semilattice.

Definition 6.0.2. A *pointed dcpo* or *cpo* is a dcpo $(P, \leq)$ with a least element $\bot \in P$.

Remark. Remember what we are after, spaces that are naturally λ-modules. Also we are expecting that, using Y, any function arising from a λ-term has a fixed point and therefore we are looking for spaces whose nice functions all have fixed points. The following proposition shows that we are on the right track.

Proposition 6.0.3. Let P be a pointed poset. Then the following are equivalent,

(a) P is a cpo

(b) every monotone map $f : P \to P$ has a last fixpoint.

6.1 Continuity

Definition 6.1.1. Let $f : P \to Q$ be a monotone map of dcpos. We say that f is,

(a) *Scott-continuous* if for every directed subset $D \subset P$ we have $f(\sup D) = \sup f(D)$

(b) *strict* if P and Q are pointed and $f(\bot) = \bot$.

We denote the poset (pointwise) of continuous functions by $[P \to Q]$ and the subposet of strict continuous functions by $[P \to Q]_\bot$.

Proposition 6.1.2. If P and Q are (pointed) dcpos then $[P \to Q]$ ($[P \to Q]_\bot$) is a (pointed) dcpos where suprema are computed pointwise.

Proof. Let $D \subset [P \to Q]$ be directed. By definition $A_x = \{f(x) \mid f \in D\}$ is directed so the function,

$$g(x) = \sup A_x = \sup_{f \in D} f(x)$$

is well-defined. Then for any directed $A \subset P$ notice that,

$$g(\sup A) = \sup_{f \in D} f(\sup A) = \sup_{f \in D} \sup_{x \in A} f(x) = \sup_{x \in A} \sup_{f \in D} f(x) = \sup_{x \in A} g(x)$$

proving that g is continuous. It is clear that g is the supremum of D. □

6.2 The Scott Topology

Definition 6.2.1. Let P be a dcpos a subset $C \subset P$ is *(Scott) closed* if it is downward and closed under suprema of directed subsets. The *Scott topology* has these as closed sets.

Proposition 6.2.2. A map $f : P \to Q$ of dcpos is Scott-continuous if and only if it is continuous in the Scott topology.

Proof. Assume f is Scott-continuous and $C \subset Q$ is closed. Then $f^{-1}(C)$ is downward because f is monotone and if $D \subset f^{-1}(C)$ is directed then $f(\sup D) = \sup f(D) \in C$ so $\sup D \in f^{-1}(C)$.

Conversely, if f is continuous for the Scott topology. For any $x \in P$ the set $D_x = \{y \in P \mid y \leq x\}$ is closed and if $x' \leq x$ then $x \in f^{-1}(D_{f(x)}$ so $x' \in f^{-1}(D_{f(x)})$ since it is closed thus $f(x') \in D_{f(x)}$ so $f(x') \leq f(x)$ so f is monotone. Furthermore, let $D \subset P$ be directed. Then $f^{-1}(D_{\sup f(D)})$ is closed and $D \subset f^{-1}(D_{\sup f(D)})$ so $\sup D \in f^{-1}(D_{\sup f(D)})$ so $f(\sup D) \leq \sup f(D)$ but also $f(D) \leq f(\sup D)$ so $f(\sup D) = \sup f(D)$. □

6.3 Fixed Points

When you first learn about functions and fixpoints you might notice that sometimes to find a fixpoint you can just iterate f. Take an arbitrary x_0 and then consider the sequence $f(x_0), f^2(x_0), f^3(x_0), \ldots$. If this converges it is guaranteed to be a fixpoint. For example, when I was bored in highschool, I took a random number on my calculator and hit cos over and over. It allways approaches $0.739085\cdots$ the cosine fixpoint. It turns out, for continuous f, this process always converges in a cpo.

Theorem 6.3.1 (Kleene)**.** Let D be a cpo,

(a) every continuous $f : D \to D$ has a least fixpoint,

$$\mathsf{fix}(f) = \sup_{n \in \mathbb{N}} f^n(\bot)$$

(b) the map $\mathsf{fix} : [D \to D] \to D$ is continuous.

Proof. The set $\{f^n(\bot)\}_{n \in \mathbb{N}}$ is a chain by monotonicity. Thus by completeness,

$$f(\sup_{n \in \mathbb{N}} f^n(\bot)) = \sup_{n \in \mathbb{N}} f(f(n)) = \sup_{n \in \mathbb{N}} f^{n+1}(\bot) = \sup_{n \in \mathbb{N}} f^n(\bot)$$

so $\mathsf{fix}(f)$ is a fixpoint of f. Let $x \in D$ be any other fixpoint. By monotonicity $f^n(\bot) \leq f^n(x) = x$ and thus $\sup_{n \in \mathbb{N}} f^n(\bot) \leq x$.

(DONT DO THE SECOND PART) Now we address the continuity of fix. Notice that $\mathsf{fix} = \sup_{n\in\mathbb{N}} \mathsf{it}_n$ where $\mathsf{it}_n : f \mapsto f^n(\bot)$. Since $[[D \to D] \to D]$ is a dcpo it suffices to show that it_n is continuous. We proceed by induction. it_0 is constant so this is continuous. Then for $F \subset [D \to D]$ directed, consider,

$$\begin{aligned}\mathsf{it}_{n+1}(\sup F) &= (\sup F)(\mathsf{it}_n(\sup F)) = (\sup F)(\sup_{f\in F} \mathsf{it}_n(f)) \\ &= \sup_{g\in F} g(\sup_{f\in F}(\mathsf{it}_n(f))) = \sup_{g\in F}\sup_{F\in F} g(\mathsf{it}_n(f)) = \sup_{f\in F}\sup_{g\in F} g(\mathsf{it}_n(f)) \\ &= \sup_{f\in F} f(\mathsf{it}_n(f))\end{aligned}$$

because the map $(f, g) \mapsto g(\mathsf{it}_n(f))$ is monotone. □

6.4 Scott Domains

6.5 Scott's Model D_∞

To build a λ-model, we start with the trivial cpo $\mathbb{N}^+$ which is $\mathbb{N}$ with the trivial order adjoin $\bot$. We think of these elements as the Church numerals along with a symbol for "ill-defined" or "DNE". Then we get an interpretation of λ-terms as follows. If $M =_\beta \underline{n}$ then we send $M \mapsto n$ and otherwise $M \mapsto \bot$. This is well-defined because Church numerals are normal so no two can be β-equivalence by the Church-Roser theorem. However, this alone is not a λ-model, there is no composition law! What we need to do, is glue in the continuous functions $[\mathbb{N}^+ \to \mathbb{N}^+]$ so we can apply them to our elements $\mathbb{N}^+$. We then interpret a λ-term as $M \mapsto f$ for $f \in [\mathbb{N}^+ \to \mathbb{N}^+]$ such that for each $n \in \mathbb{N}$ we have $(M\ \underline{n}) =_\beta \underline{f(n)}$ and $f(n) = \bot$ if $(M\ \underline{n})$ does not reduce to a Church numeral. Finally, I need to say what $f(\bot)$ is. To impose continuity $f(\bot) = \bot$ unless $f|_{\mathbb{N}}$ is constant and then $f(\bot) = f(\mathbb{N})$. Howver, we have not completed a λ-model because we don't know how to apply integer functions to eachother or how to actually do this "gluing". The "building-up" process of cpos is accomplished by the following construction,

Definition 6.5.1. Let D, D' be cpos. A *projection* from $D' \to D$ is a pair (ϕ, ψ) of continuous maps $\phi : D \to D'$ and $\psi : D' \to D$ such that,

$$\psi \circ \phi = \mathrm{id}_D \quad \text{and} \quad \phi \circ \psi \leq \mathrm{id}_{D'}$$

Now we construct a sequence of cpos inductively. Let $D_0 = \mathbb{N}^+$ (adjoin $\bot$ to $\mathbb{N}$ equipped with the trivial order) then define $D_{n+1} = [D_n \to D_n]$. We construct projections $D_{n+1} \to D_n$ inductively as follows. Let $\phi_0(d) = \kappa_d$ where κ is the constant function and $\psi_0(c) = c(\top)$. Then we define $\phi_n : D_n \to D_{n+1}$ and $\psi_n : D_{n+1} \to D_n$,

$$\phi_n(\sigma) = \phi_{n-1} \circ \sigma \circ \psi_{n-1} \quad \text{and} \quad \psi_n(\tau) = \psi_{n-1} \circ \tau \circ \phi_{n-1}$$

It is not difficult to show that this gives a sequence of projections. Then we define Scott's model,

$$D_\infty = \varinjlim_n D_n$$

Explicitly the elements are sequences $(d_0, d_1, d_2, \dots)$ such that $\psi_n(d_{n+1}) = d_n$ for all n with the ordering pointwise.

Proposition 6.5.2. D_∞ is a cpo with $\bot = (\bot_0, \bot_1, \bot_2, \dots)$ and if $A \subset D_\infty$ is directed then,

$$\sup X = (\sup X_0, \sup X_1, \sup X_2, \dots)$$

and there are projections $D_\infty \to D_n$ with $\phi_n : d \mapsto d_n$ and corresponding inclusion $\psi_n : d \mapsto (\psi_{n,0}(d), \psi_{n,1}(d), \psi_{n,2}(d), \dots)$.

We can then define the composition on D_∞ as,

$$a \bullet b = \sup_n \psi_n(a_{n+1}(b_n))$$

Proposition 6.5.3. The composition gives an isomorphism $D_\infty \xrightarrow{\sim} [D_\infty \to D_\infty]$.

Then we can show there is a natural way to make $(D_\infty, \bullet)$ into a λ-model extending the interpretation we gave for the first two stages. How are we supposed to think about elements of D_∞. The relation $\leq$ is expressing the idea of "defined on a larger set". Indeed, for $f, g \in D_1$ we have $f \leq g$ iff $f(n) = g(n)$ if $f(n) \neq \bot$ and whenever $g(n) = \bot$ then $f(n) = \bot$ meaning g is defined at least everywhere f is and on the domain where f is defined the two functions agree. Then the global bottom element $\bot \in D_\infty$ represents the function defined nowhere. Thefore, our supremum processes can be thought of as producing the limit of a sequence of functions that are getting progressively defined more often.

6.6 D_∞ is a λ-model

THINK OF THIS AS COMPUTABLE APPROXIMATION

6.7 Fixpoints in D_∞

Using our interpretation of D_∞ recall that the fixpoint operator takes the form,

$$\mathsf{fix}(f) = \sup_{n \in \mathbb{N}} f^n(\bot)$$

Consider the recursion operator T we defined before. Remember that $\mathsf{fix}(T)$ is a solution to the self-referential recursive definition. Explicitly, we get a sequence of functions,

$$\bot, T(\bot), T^2(\bot), T^3(\bot), \dots$$

What do these mean? The first is just defined nowhere. The second is the function (when applied to Church numerals),

$$T(\bot)(x) = \begin{cases} n & x = 0 \\ \bot & x > 0 \end{cases}$$

so it is defined just at $x = 0$ to equal the base case. Then,

$$T^2(\bot)(x) = \begin{cases} n & x = 0 \\ G(n, 1) & x = 1 \\ \bot & x > 1 \end{cases}$$

and so on. We see that this is converging to the expected behavior of the recursive function.

However, recall that we have a different mysterious way of producing fixed points, the operator Y. In Scott's model $Y = \mathsf{fix}$ but this does not always happen.

Remark. Using the $D_\infty \to [D_\infty \to D_\infty]$ is an isomorphism we can regard $\mathsf{fix} \in D_\infty$.

Proposition 6.7.1. In D_∞ we have $Y = \mathsf{fix}$.

Proof. It is clear that $\mathsf{fix} \leq Y$ since fix gives the smallest fixpoint. Conversely, for $x \in D$ we have,

$$Y \bullet x = X \bullet X$$

where X is such that $X \bullet d = x \bullet (y \bullet y)$. Then by definition,

$$X \bullet X = \sup_{n \in \mathbb{N}} \psi_n(X_{n+1}(X_n))$$

□

7 The Simply-Typed λ-Calculus

7.1 Expressible Functions

7.2 Models

7.3 Curry-Howard

8 References

(a) https://www.youtube.com/watch?v=7cPtCpyBPNI

(b) http://www.cs.nott.ac.uk/ pszgmh/dom4.pdf

(c) https://en.wikipedia.org/wiki/Kleene$_f ixed - point_t heorem$