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. Think of $\lambda x.M$ as the function taking in x and returing 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 $(M\,N)$ 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 ν-*redex* and its ν-*reduction* is the corresponding term $M[x := N]$. We write,

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

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

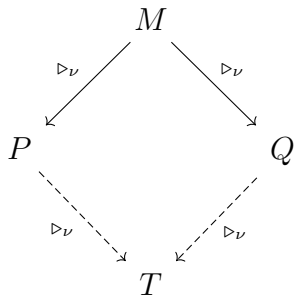
$$M \triangleright_{\nu} N$$

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

Example 1.0.3. $(\lambda x.x)y \triangleright_{\nu,1} y$ 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 ν-redices. We say that N *is a normal form* of M if N is a normal form and $M \triangleright_{\nu} N$.

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



Corollary 1.0.6. Suppose M, N are λ-terms with $M =_\nu N$ then there exists a λ-term T such that $M \triangleright_\nu T$ and $N \triangleright_\nu T$. 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,

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

and there are no other possible ν-reductions. Therefore Ω does not have a ν-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 $n \in \mathbb{N}$,

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

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,

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

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

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,

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

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 ν-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 least 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,

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

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 $\sup_{n \in \mathbb{N}} f^n(\bot) \leq x$.

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

Definition 6.4.1. Let P be a partially ordered set. Then $x \in P$ is a *compact element* if for every directed subset $D \subset P$ with $x \leq \sup D$ then $x \leq d$ for some $d \in D$.

Remark. If we think of the partial order P as a category then sup of a directed subset is exactly the filtered colimit. Thus compact elements of P are exactly the compact objects of this category.

Definition 6.4.2. A nonemtpy dcpo D is a *Scott domain* if D is

(a) *bounded-complete*, i.e. all bounded subsets of D have a supremum

(b) *algebraic* or *compactly-generated*, i.e. every element of D is obtained as the supremum of a directed set of compact elements of D.

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 $M =_\nu \underline{n}$ 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 ν-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 $(M\ \underline{n}) =_\nu \underline{f(n)}$ 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 $X \subset D_\infty$ 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 $D_\infty \xrightarrow{\sim} [D_\infty \to D_\infty]$ 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 $x \in D_\infty$ we have,

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

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

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

FINISH THIS Look here The Y-combinator in Scott's lambda-calculus models. □

7 The Simply-Typed λ-Calculus

Untyped λ-calculus has a simplictic beauty and is maximally powerful. However, this power and flexibility comes at a cost as Church originally found when is logical system built from the untyped calculus was found to contain contradictions.

Adding types to the language is advantangeous from two perspectives. From the perspectives of logic, it allows the creation of a well-founded and consistent system that avoids paradoxes by building up terms hierachically (hence excising self-references in the way that achieved by Russel and Whitehead in *Principia Mathematica*). From the perspective of programing language theory, types constrain functions *formally* and therefore aid in the design and interpretation of programs, languages, and compilers. Futhermore, it is usually mathematically sensible to constrain the domain of definitions of functions otherwise their definition will require significantly more work and confusion or one will be awash in undefined behavior.

7.1 Simply Typed Lambda Terms

Definition 7.1.1. A *simply typed* λ*-calculus* constructed on a set of *atomic types* Σ consists of a set of types defined by the inductive rule

(a) 1 is a type

(b) any atomic type $\alpha \in \Sigma$ is a type

(c) if α, ν are types then $\alpha \times \nu$ is a type

(d) if α, ν are types then $\alpha \to \nu$ is a type

for each type σ there is an infinite set of variables V_σ of type σ write $x : \sigma$ for $x \in V_\sigma$. A *context* or *environment* Γ is a finite list of typed variables. We say M is a *term* of type α in Γ if the typing judgement $\Gamma \vdash M : \alpha$ can be derived through the rules:

$$\frac{}{\Gamma \vdash \star : 1}\ \text{UNIT}$$

$$\frac{}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{VAR}$$

$$\frac{\Gamma, x : \alpha \vdash M : \nu}{\Gamma \vdash (\lambda x : \alpha.M) : \alpha \to \nu}\ \text{ABS}$$

$$\frac{\Gamma \vdash M : \alpha \to \nu \qquad \Gamma \vdash N : \alpha}{\Gamma \vdash (M\,N) : \nu}\ \text{APP}$$

$$\frac{\Gamma \vdash M : \alpha \qquad \Gamma \vdash N : \alpha}{\Gamma \vdash \langle M, N \rangle : \alpha \times \nu}\ \text{PAIR}$$

Theorem 7.1.2. If $\Gamma \vdash M : \alpha$ and $\Gamma \vdash M : \nu$ then $\alpha = \nu$.

Then we define the main operations on terms: α-converstion, ν-reduction, and η-reduction.

Definition 7.1.3. Given $(\lambda x : \sigma.M)$ and $y : \sigma$ is a variable not appearing in M then we write $\lambda x : \sigma.M \equiv_\alpha \lambda y : \sigma.M[x := y]$ are α-equivalent or *equivalent via* α*-conversion.* Furthermore, if $\Gamma \vdash M : 1$ then $M \equiv_\alpha \star$.

Proposition 7.1.4. Typing judgements respect α-equivalence:

$$\Gamma \vdash (\lambda x : \sigma.M) : \tau \iff \Gamma \vdash (\lambda y : \sigma.M[x := y]) : \tau$$

Definition 7.1.5. A λ-term of the form $(\lambda x.M)\,N$ is called a *ν-redex* and its *ν-reduction* is the corresponding term $M[x := N]$ (where we α-convert any variables in N to not clash with variables in M). We write,

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

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

$$M \triangleright_{\nu} N$$

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

Proposition 7.1.6. Suppose that $\Gamma \vdash (\lambda x : \sigma : M) : \sigma \to \tau$ and $\Gamma \vdash N : \tau$ then $\Gamma \vdash M[x := N] : \tau$ therefore if $\Gamma \vdash M : \sigma$ and $M \triangleright_{\nu} N$ then $\Gamma \vdash N : \sigma$ so ν-reduction preserves typing judgements.

Definition 7.1.7. A term is *normal* if it contains no ν-redexes.

Theorem 7.1.8. Starting from any term M chosing any sequence of ν-reductions produces a normal form independent of the sequence up to α-equivalence.

7.2 The Categorical Semantics

Definition 7.2.1. Let Λ be a simply-typed λ-calculus (meaning we introduce symbols of fixed terms and equations) then we form a category C_Λ whose objects are the types of Λ and whose morphisms

$$\mathrm{Hom}_{C_\Lambda}(\alpha, \nu) = \{M \mid \text{closed terms } \vdash M : \alpha \to \nu\}/ =_{\alpha\nu\eta}$$

Composition for $N : \alpha \to \nu$ and $M : \nu \to \gamma$ is the equivalence class of the term $(M\,N) : \alpha \to \gamma$.

Proposition 7.2.2. C_Λ is a Cartesian closed category.

Definition 7.2.3. Let C be a Cartesian closed category. Then there is an associated simply typed λ-calculus called the *internal language*. Its types are the objects of C. For each $\sigma \in C$ we define an infinite list of variables V_σ and write $x : \sigma$ if $x \in V_\sigma$. For each morphism $f : \alpha \to \nu$ we introduce a new symbol f of type $\alpha \to \nu$ and we define the reduction of $(g\,f) := g \circ f$.

Theorem 7.2.4 (Lambeck-Scott)**.** These functors form an equivalence between λ-calculi and Cartesian closed categories.

7.3 Expressible Functions

7.4 Models

8 Curry-Howard

The Curry-Howard principle says that *proof theory* and the *theory of computation* are two viewpoints on the same mathematical object – called *proofs* in one and *programs* in the other.

The first incarnation of this correspondence is the Brouwer-Heytin-Kolmogorov interpretation of intuitionistic logic: that proofs are "procedures" explicitly that a proof of $A \to B$ is a "transformation" or "procedure" or – in modern language – a program that takes proofs of A to proofs of B. This is a transformational step in the developement of the viewpoint of *proofs as mathematical objects*.

8.1 Curry-Howard-Lambeck for simply typed λ-calculus

Here we follow these notes.

Definition 8.1.1. The sequence calculus for the fragment of intuitionistic logic with connectives $(\wedge, \to)$ has a set of formulas defined as follows:

(a) propositional variables are formulas

(b) if A and B are formulas then $A \wedge B$ and $A \to B$ are formulas.

A *context* Γ is a list of formulas. A *judgement* is a setence of the form $\Gamma \vdash A$ read Γ *proves* A that is inductively built from the following structural rules:

$$\frac{}{\Gamma, A \vdash A}\ \text{AX} \qquad \frac{\Gamma_1, A, B, \Delta \vdash C}{\Gamma_1, B, A, \Delta \vdash C}\ \text{EXC}$$

$$\frac{\Gamma, A, A, \Delta \vdash B}{\Gamma, A, \Delta \vdash B}\ \text{CTR} \qquad \frac{\Gamma \vdash B}{\Gamma, A \vdash B}\ \text{WKN}$$

and the following logical rules:

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A \wedge B, \Delta \vdash C}\ \wedge_L \qquad \frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \wedge B}\ \wedge_R$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \to_R \qquad \frac{\Gamma \vdash A \qquad B, \Delta \vdash C}{\Gamma, A \to B, \Delta \vdash C}\ \to_L$$

Example 8.1.2. Lets show how to prove things like $\Gamma, A \wedge B \vdash A$.

$$\dfrac{\dfrac{\dfrac{}{\Gamma, A \vdash A}\ \text{AX}}{\Gamma, A, B \vdash A}\ \text{WKN}}{\Gamma, A \wedge B \vdash A}\ \wedge_L$$

The advantage of natural deduction systems over Hilbert style systems (like those two-column proofs you had to write in high school) is that the context is allowed to change throughout the deduction. In a Hilbert style system every line must be a theorem of the given global context (i.e. each line must be true) while in natural deduction we are allowed to introduce hypotheticals into the context and deduce sentences that are only true in context.

Example 8.1.3. Here is a proof of $(A \to B) \wedge (B \to C) \vdash A \to C$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\ \text{AX} \quad \dfrac{}{B \vdash B}\ \text{AX}}{A, A \to B \vdash B}\ \to_L \quad \dfrac{\dfrac{}{B \vdash B}\ \text{AX} \quad \dfrac{}{C \vdash C}\ \text{AX}}{B, B \to C \vdash B}\ \to_L}{A, A \to B, B \to C \vdash C}\ \to_L}{A \to B, B \to C \vdash A \to C}\ \to_R}{(A \to B) \wedge (B \to C) \vdash A \to C}\ \wedge_L$$

Definition 8.1.4 (Cut rule)**.** Consider the rule

$$\frac{\Gamma \vdash A \qquad \Delta, A \vdash B}{\Gamma, \Delta \vdash B}\ \text{CUT}$$

Theorem 8.1.5 (Cut-elimination)**.** Any theoem proved using CUT can be proved without it.

We think of cut elimination as executing "computation" in our logical system. If a proof is a program, then the process of simplifying a proof via cut elimination is analogous to computation via ν-reduction as a "simplification" process on λ-terms.

Definition 8.1.6. A category C is *Cartesian closed* if it has all finite products and there is a right-adjoint to $(-) \times X$ called *internal Hom* or the *exponential object* denoted as $[X \to (-)]$ or sometimes $(-)^X$.

Definition 8.1.7. Let C be a Cartesian closed category. An interpretation of the above logic in C is an assignment of an object $[\![A]\!]$ for each formula A such that,

$$[\![A \wedge B]\!] := [\![A]\!] \times [\![B]\!] \quad [\![A \to B]\!] := [[\![A]\!] \to [\![B]\!]]$$

Each judgment $\Gamma \vdash A$ is interpreted as claiming the existence of a morphism $f : [\![\Gamma]\!] \to [\![A]\!]$ where $[\![\Gamma]\!] := \prod_{B \in \Gamma} [\![B]\!]$ but the construction of the morphism *depends on the proof.* The deduction rules are interpreted as the following operations on morphisms

$$\frac{}{\Gamma, A \vdash A}\ \text{AX}$$

$$\mathrm{id}_{[\![A]\!]} : [\![A]\!] \to [\![A]\!]$$

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}\ \text{EXT}$$

$$\text{compose with } [\![A]\!] \times [\![B]\!] \to [\![B]\!] \to [\![A]\!]$$

$$\frac{\Gamma, A, A, \Delta \vdash B}{\Gamma, A, \Delta \vdash B}\ \text{CTR}$$

$$\text{compose with } \Delta : [\![A]\!] \to [\![A]\!] \times [\![A]\!]$$

$$\frac{\Gamma \vdash B}{\Gamma, A \vdash B}\ \text{WKN}$$

$$\text{compose with } \pi_1 : [\![\Gamma]\!] \times [\![A]\!] \to [\![\Gamma]\!]$$

and the following logical rules:

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A \wedge B, \Delta \vdash C}\ \wedge_L$$

$$\text{nothing}$$

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \wedge B}\ \wedge_R$$

$$\text{universal property of } [\![A]\!] \times [\![B]\!]$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \to_R$$

$$\text{adjunction: } \lambda : \mathrm{Hom}_C([\![\Gamma]\!] \times [\![A]\!], [\![B]\!]) \to \mathrm{Hom}_C([\![\Gamma]\!], [[\![A]\!] \to [\![B]\!]])$$

$$\frac{\Gamma \vdash A \qquad B, \Delta \vdash C}{\Gamma, A \to B, \Delta \vdash C}\ \to_L$$

$$\text{compose with evaluation map } \mathrm{ev} : [\![A]\!] \times [[\![A]\!] \to [\![B]\!]] \to [\![B]\!]$$

Theorem 8.1.8. The interpretations of proofs modulo cut elimination are well-defined.

Proof. This is exactly the adjoint relations $\mathrm{ev} \circ (\lambda f \times \mathrm{id}_Y) = f$ for any $f : X \times Y \to Z$ and $\lambda(\mathrm{ev} \circ (h \times \mathrm{id}_Y)) = h$ for any $h : X \to [Y \to Z]$. □

9 Linear Logic

9.1 Differentiation

We define

$$\frac{!A \vdash B}{!A, A \vdash B}\ \text{DIFF}$$

via the composition

$$\frac{\dfrac{!A \vdash B}{!A, !A \vdash B}\ \text{COCTR}}{!A, A \vdash B}\ \text{CODER}$$

10 References

(a)

(b)

(c) klenn fix point

(d) amar notes

11 Linear Logic

We want a logic which can be interpreted in the category of vectorspaces instead of in the category of sets. The problem is that to interpret standard intutitionistic logic (or simply-typed λ-calculus) we need a Cartesian closed category. This rules out any Abelian category unless we modify the logic sufficiently. This is what we hope to accomplish with linear logic. There are a number of other interpretations of linear logic,

Interpretations:

(a) in order to have linear (interpretable in an abelian category) semantics we need to give up exponential objects

(b) a logic that mirrors quantum mechanics (can be interpreted in the category of Hilbert spaces). One can think of the no cloning theorem as an illustration of the nonlinearity of classical logic and the linearity of quantum logic. Precisiely, there does not exist a *linear* diagonal map $\mathcal{H} \to \mathcal{H} \otimes \mathcal{H}$ which would correspond to duplication

(c) Linear logic captures reasoning where it is important to treat the hypotheses as resources which are not infinitely expendable. For example, from the sentences

   (a) "if I have five dollars I can buy a coffee"

   (b) "if I have five dollars I can buy a sandwich"

   it seems that the following deduction rule:

$$\frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A \to C}{\Gamma \vdash A \to B \wedge C}$$

   would allow us to make the fallacious reasoning

   "if I have five dollars I can buy a coffee and a sandich"

In this case, we need to consider the hypothesis $A =$ "if I have 5 dollars" as an expendable resource and it is important to track how many times a hypothesis is used in a proof in order to make valid deductions. Heuristically, we say that such proofs must be "linear" in the sense that they only use each hypothesis one so any formulas one might write down can be only first order.

This is an interesting idea but it seems that such a logic will not be very expressive. For example,

(a) the Church encoding of the numeral 2

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

is nonlinear in f in an essential way.

If we want ot be able to do anything like arithmetic in this logic, we need to have some mechanism for dealiing with nonlinearity. The essential idea is that of "exponentiation" we will allow some hypothesis A to be used arbitrarily many times but only if they are modified by the "exponential" operator $!A$. We will not present a fragment of *intuitionistic linear logic*. The language is given by infinitely many propositional variables $x, y, z, \ldots$ and formed inductively via two binary connectives $\multimap$ and $\otimes$ and one unary connective $!$. There is a constant 1. The set of *formlas* is defined inductively as follows: any varaible or constant is a formula and if A, B are formulas then

(a) $A \multimap B$

(b) $A \otimes B$

(c) $!A$

are formulas. The reason for the new symbol $\otimes$ is that it will be interpreted as tensor product in the linear semantics. The new symbol $\multimap$ is just to psychologically distinguish the conditional in linear logic from the material conditional of ordinary intuitionistic logic. In fact, $A \to B$ will turn out to be closer in meaning to $!A \multimap B$ than $A \multimap B$.

11.1 Deduction Rules

$$\frac{}{\Gamma, A \vdash A}\ \text{AX} \qquad \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}\ \text{EXC}$$

$$\frac{\Gamma, !A, !A, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\ \text{CTR} \qquad \frac{\Gamma \vdash B}{\Gamma, A \vdash B}\ \text{WKN}$$

Notice that contraction is only allowed for exponentiated objects. This is because linear proofs using a standard hypothesis twice cannot automatically be reduced to using the hypothesis once. There are also the following logical rules:

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A \otimes B, \Delta \vdash C}\ \otimes_L \qquad \frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\ \otimes_R$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\ \multimap_R \qquad \frac{\Gamma \vdash A \qquad \Delta, B \vdash C}{\Gamma, \Delta, A \multimap B \vdash C}\ \multimap_L$$

there are then the logical rules relating to exponentiation

$$\frac{!\Gamma \vdash A}{!\Gamma \vdash !A}\,\text{PRO} \qquad\qquad \frac{\Gamma A, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\,\text{DER}$$

there are also some obvious rules involving 1

$$\frac{\Gamma, \Delta \vdash A}{\Gamma, 1, \Delta \vdash A}\,\text{1-L} \qquad\qquad \frac{}{\vdash 1}\,\text{1-R}$$

Example 11.1.1. Note that $!A \vdash A \otimes A$ but $A \nvdash A \otimes A$. Indeed, consider the following proof

$$\frac{\dfrac{}{A \vdash A}\,\text{AX} \quad \dfrac{}{A \vdash A}\,\text{AX}}{A, A \vdash A \otimes A}\,\otimes_R$$

but notice we cannot apply contraction to get $A \vdash A \otimes A$. Indeed this corresponds to the fact that there is no linear diagonal map $V \to V \to V$ but given $\alpha \in V$ and $\beta \in V$ there is $\alpha \otimes \beta \in V \otimes V$. However, if we derelict A then we can make the proof go through

$$\frac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,\text{AX}}{!A \vdash A}\,\text{DER} \quad \dfrac{\dfrac{}{A \vdash A}\,\text{AX}}{!A \vdash A}\,\text{DER}}{!A, !A \vdash A \otimes A}\,\otimes_R}{!A \vdash A \otimes A}\,\text{CTR}$$

Example 11.1.2. For any fomula A let

$$\mathbf{int}_A := !(A \multimap A) \multimap (A \multimap A)$$

by the type of integers valued in A. This exactly corresponds to the Church encoding of the integers as functions that take a function $f : A \to A$ and return a function $A \to A$ which is the n^{th}-iterate of f. However, note that in linear logic we need to exponentiate the hypothesis $(A \to A)$ in order to iterate f. Recall, that under the Curry-Howard correspondence, functions of type $\mathbf{int}_A$ correspond to proofs $\vdash \mathbf{int}_A$. The object $\underline{1}_A$ is the following proof of $\mathbf{int}_A$

$$\frac{\dfrac{\dfrac{}{A \multimap A \vdash A \multimap A}\,\text{AX}}{!(A \multimap A) \vdash A \multimap A}\,\text{DER}}{\vdash !(A \multimap A) \multimap (A \multimap A)}\,\multimap_R$$

whereas $\underline{0}_A$ is the following proof

$$\frac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,\text{AX}}{\vdash A \multimap A}\,\multimap_R}{!(A \multimap A) \vdash A \multimap A}\,\text{WKN}}{\vdash !(A \multimap A) \multimap (A \multimap A)}\,\multimap_R$$

Finally, the proof $\underline{2}_A$ is a bit more complicated,

$$\frac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A} \quad \dfrac{\dfrac{}{A \vdash A} \quad \dfrac{}{A \vdash A}}{A, A \multimap A \vdash A}\,\multimap_L}{A, A \multimap A, A \multimap A \vdash A}\,\multimap_L}{A \multimap A, A \multimap A \vdash A \multimap A}\,\multimap_R}{!(A \multimap A), A \multimap A \vdash A \multimap A}\,\text{DER}}{!(A \multimap A), !(A \multimap A) \vdash A \multimap A}\,\text{DER}}{!(A \multimap A) \vdash A \multimap A}\,\text{CTR}}{\vdash !(A \multimap A) \multimap (A \multimap A)}\,\multimap_R$$

To understand why this corresponds to the number 2 we need to understand cut elimination.

Definition 11.1.3 (Cut rule)**.** Consider the rule

$$\frac{\Gamma \vdash A \qquad \Delta, A \vdash B}{\Gamma, \Delta \vdash B}\ \text{CUT}$$

Theorem 11.1.4 (Cut-elimination)**.** Any theoem proved using CUT can be proved without it.

Cut-elimination really says something much more interesting about the structure of the proofs themselves. The real content is in giving a sequence of elementary cut-elimination rules on profs allowing cuts and prove that after a finite number of steps we end up at a cut-free proof. These elementary reductions are intended to mirror ν-reduction via the Curry-Howard correspondence or alternatively they are the fundamental adjunction relations in the Categorical semantics we will now explore.

Definition 11.1.5. Let $\Gamma \vdash A$ be a judgement in linear logic and $\mathcal{P}$ the set of proofs. There is a birary relation $\rho \triangleright_{\text{cut}} \pi$ if π is an immediate reduction of ρ by an elementary cut-elimination rule. We refer to the equivalence relation $=_{\text{cut}}$ generated by $\triangleright_{\text{cut}}$ as *cut-equivalence* and the steps *cut-elimination transformations*.

Example 11.1.6. Let π be any proof of $A \vdash A$ then the proof of $A \multimap A$ denoted $\underline{2}_A \mid \text{prom}(\pi)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,\pi}{\vdash A \multimap A}\multimap_R}{\vdash !(A \multimap A)}\,\text{PRO} \qquad \dfrac{}{!(A \multimap A) \vdash A \multimap A}\,\underline{2}_A}{\vdash A \multimap A}\,\text{CUT}$$

is cut equivalent to the proof $\pi \mid \pi$

$$\dfrac{\dfrac{\dfrac{}{A \vdash A}\,\pi \qquad \dfrac{}{A \vdash A}\,\pi}{A \vdash A}\,\text{CUT}}{\vdash A \multimap A}\multimap_R$$

which represents the composition of π with itself.

11.2 Linear Semantics

Let k be a field why in the notes algebraically closed and of characteristic zero.

Definition 11.2.1. A *denotation* is an assignment $A \mapsto [\![A]\!]$ from sentences to k-vectorspaces defined inductively

(a) the propositional variabesl are assigned to why in the notes finite dimensional? vectorspaces

(b) $[\![1]\!] = k$

(c) $[\![A \otimes B]\!] = [\![A]\!] \otimes [\![B]\!]$

(d) $[\![A \multimap B]\!] = [[\![A]\!] \to [\![B]\!]]$

(e) $[\![!A]\!] = ![\![A]\!]$ where $!V$ is the cofree coalgebra.

Furthermore, there are denotational rules for proofs as morphisms parallel to our previous construction. We will only write down the new rules corresponding to the exponential

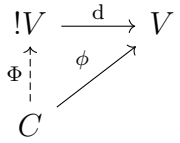
$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \wedge B} \wedge_R$$

universal property of $[\![A]\!] \times [\![B]\!]$

$$\frac{\Gamma \vdash A \qquad B, \Delta \vdash C}{\Gamma, A \to B, \Delta \vdash C} \to_L$$

compose with evaluation map
$\mathrm{ev} : [\![A]\!] \times [[\![A]\!] \to [\![B]\!]] \to [\![B]\!]$

Definition 11.2.2. The *cofree coalgebra* $!V$ is the universal coalgebra equipped with a map $\mathrm{d} : !V \to V$ universal in the sense that given any linear map $\phi : C \to V$ from a coalgebra there is a unique morphism of coalgebras $\Phi : C \to !V$ such that



If V is a finite-dimensional vector space then

$$!V = \bigoplus_{v \in V} \mathrm{Sym}\,(V)\,|\varnothing\rangle_v$$

we write $|v_1, \ldots, v_n\rangle_v := v_1, \ldots, v_n\,|\varnothing\rangle_v$ then the comultiplication structure is

$$!V \to !V \otimes !V \quad v_1, \ldots, v_{nv} \mapsto \sum_{I \subset \{1,\ldots,n\}} |v_I\rangle_v \otimes |v_{I^c}\rangle_v$$

and there is a canonical linear map

$$\mathrm{d} : !V \to V \quad |\varnothing\rangle_v \mapsto v$$

and all other $v_1 \otimes \cdots \otimes v_n\,|\varnothing\rangle_v \mapsto 0$.

Remark. Notice that there is a nonlinear map $V \to !V$ given by $v \mapsto |\varnothing\rangle_v$ this is important for the interpretation of morphisms arising from proofs involving dereliction.

Example 11.2.3. Let A be a formula and $A = [\![A]\!]$. The denotation of the proof $\underline{2}_A$ of $\vdash \mathbf{int}_A$ is the morphism

$$[\![\underline{2}_A]\!] : k \to \mathrm{Hom}_k\left(!\mathrm{End}_k\,(V)\,, \mathrm{End}_k\,(V)\right)$$

which is the linear map

$$!\mathrm{End}_k\,(V) \xrightarrow{\Delta} !\mathrm{End}_k\,(V) \otimes !\mathrm{End}_k\,(V) \xrightarrow{\mathrm{d}\otimes\mathrm{d}} \mathrm{End}_k\,(V)^{\otimes 2} \xrightarrow{-\circ-} \mathrm{End}_k\,(V)$$

Therefore $[\![\underline{2}_A]\!]$ is the map

$$|\varnothing\rangle_\alpha \mapsto |\varnothing\rangle_\alpha \otimes |\varnothing\rangle_\alpha \mapsto \alpha \otimes \alpha \mapsto \alpha \circ \alpha$$

11.3 Derivatives

Let's investigate a bit more what the map $[\![\underline{2}_A]\!]$ does. Indeed, consider its action on elements of the form $|\nu\rangle_\alpha$ where $\nu, \alpha \in \mathrm{End}_k(V)$. Indeed,

$$|\nu\rangle_\alpha \mapsto |\nu\rangle_\alpha \otimes |\varnothing\rangle_\alpha + |\varnothing\rangle_\alpha \otimes |\nu\rangle_\alpha \mapsto \nu \otimes \alpha + \alpha \otimes \nu \mapsto \nu \circ \alpha + \alpha \circ \nu = \{\nu, \alpha\}$$

which looks like the derivative of the squaring map

$$\alpha \mapsto \alpha \circ \alpha$$

at α in the direction ν. This interpretation suggests a definition of the derivative in general.

For each proof $\vdash \alpha : A$ there is a vector $[\![\alpha]\!] \in [\![A]\!]$ (the corresponding morphism $1 \to [\![A]\!]$) and hence a group-like element $|\varnothing\rangle_{[\![\alpha]\!]} \in ![\![A]\!]$. Likewise, givien a proof $\psi : !A \vdash B$ the denotation is a linear map

$$[\![\psi]\!] : ![\![A]\!] \to [\![B]\!]$$

It is the behavior of the denotation of $[\![\psi]\!]$ on these group-like elements which recovers the input-output behavior of ψ since

$$[\![\psi]\!] |\varnothing\rangle_{[\![\alpha]\!]} = [\![\psi(\alpha)]\!]$$

where $\psi(\alpha) : B$ is the proof of B given by applying cut-elimination to $\psi \mid \mathrm{prom}(\alpha)$. However there is more information in the linear map $[\![\psi]\!]$ than the input-output behavior of ψ. For example, we can lift $[\![\psi]\!]$ to a coalgebra morphism

$$![\![\psi]\!] : ![\![A]\!] \to ![\![B]\!]$$

and given proofs $\alpha, \beta : A$ there is an associated primitive element $|[\![\beta]\!]\rangle_{[\![\alpha]\!]} \in ![\![A]\!]$ and in fact if we take the limit formally (the output will be polynomial in h)

$$[\![\psi]\!] |[\![\beta]\!]\rangle_{[\![\alpha]\!]} = \lim_{h \to 0} \frac{[\![\psi]\!] |\varnothing\rangle_{[\![\alpha]\!] + h[\![\beta]\!]} - [\![\psi]\!] |\varnothing\rangle_{[\![\alpha]\!]}}{h}$$

There is an algebraic interpretation of this derivative. Algebraic geometry teaches us that if we consider a finite-dimensional vector space as a scheme $\underline{V} = \mathrm{Spec}(\mathrm{Sym}(V^\vee))$ then a tangent vector ξ of this scheme at a point $v \in V$ is a map

$$\mathrm{Spec}(k[\epsilon]) \to \underline{V}$$

such that $\mathrm{Spec}(k) \to \mathrm{Spec}(k[\epsilon])$ maps to v. Then pairs (v, ξ) correspond to

$$\mathrm{Hom}_{k\text{-alg}}(\mathrm{Sym}(V^\vee), k[\epsilon]) = \mathrm{Hom}_k(V^\vee, k[\epsilon]) = \mathrm{Hom}_k((k[\epsilon])^*, V) = \mathrm{Hom}_{k\text{-coalg}}((k[\epsilon])^*, !V)$$

where $k[\epsilon]$ is a Hopf algebra and thus its dual, we call $\mathcal{T}$ with basis $1, \varepsilon = \epsilon^*$, is also a coalgebra.

Lemma 11.3.1. Coalgebra morphisms $\mathcal{T} \to !V$ are always of the form $\theta(1) = |\varnothing\rangle_\alpha$ and $\theta(\varepsilon) = |u\rangle_v$ for some $v, u \in V$.

Proof. Indeed $\Delta(1) = 1 \otimes 1$ so $\theta(1)$ is grouplike and hence is of the form $|\varnothing\rangle_v$. Furthermore $\Delta(\varepsilon) = \varepsilon \otimes 1 + 1 \otimes \varepsilon$ and therefore $\theta(\epsilon)$ must be of the form $|u\rangle_v$ since these are the only elements satisfing $\Delta(|u\rangle_v) = |u\rangle_v \otimes |\varnothing\rangle_v + |\varnothing\rangle_v \otimes |u\rangle_v$. □

Definition 11.3.2. Given a proof $\vdash \alpha : A$ a *tangent vector* at π is a morphism of coalgebras $\theta : \mathcal{T} \to !\llbracket A \rrbracket$ with the property that $\theta(1) = |\varnothing\rangle_{\llbracket \pi \rrbracket}$ meaning that the diagram

$$\begin{array}{ccc} k & \xrightarrow{\llbracket \pi \rrbracket} & \llbracket A \rrbracket \\ \downarrow & & \uparrow{\scriptstyle \mathrm{d}} \\ \mathcal{T} & \xrightarrow{\theta} & !\llbracket A \rrbracket \end{array}$$

Notice that there is an isomorphism between $\llbracket A \rrbracket$ and the tangent space given by sending $Q \in \llbracket A \rrbracket$ to the coalgebra morphism $\theta : \mathcal{T} \to !\llbracket A \rrbracket$ such that

$$\theta(1) = |\varnothing\rangle_{\llbracket \pi \rrbracket} \qquad \theta(\epsilon) = |Q\rangle_{\llbracket \pi \rrbracket}$$

Note that the denotation of a program not only maps inputs to outputs but also tangent vectors to tangent vectors. Indeed, given $\llbracket \psi \rrbracket : !\llbracket A \rrbracket \to \llbracket B \rrbracket$ we lift to $!\llbracket \psi \rrbracket : !\llbracket A \rrbracket \to !\llbracket B \rrbracket$ then we get a map

$$\mathcal{T} \xrightarrow{\theta} !\llbracket A \rrbracket \xrightarrow{!\llbracket \psi \rrbracket} !\llbracket B \rrbracket$$

which is a tangent vector at the proof $\psi(\alpha) := \rho \mid \mathrm{prom}(\pi)$. This is equivalent to the following definition.

Definition 11.3.3. Given $\vdash \pi : !A \multimap B$ and proves $\vdash \alpha, \beta : A$ then we interpret $\llbracket \alpha \rrbracket, \llbracket \beta \rrbracket \in \llbracket A \rrbracket$ then we consider

$$\begin{array}{ccc} \llbracket !A \rrbracket & \xrightarrow{\llbracket \pi \rrbracket} & \llbracket B \rrbracket \\ {\scriptstyle \llbracket \alpha \rrbracket \oplus \llbracket \beta \rrbracket}\uparrow & & \uparrow \\ \mathcal{T} & \xrightarrow[\llbracket \pi(\alpha) \rrbracket \oplus \llbracket \partial_\beta \alpha \rrbracket]{} & \llbracket !B \rrbracket \end{array}$$

where the right map exists by the universal property of $\llbracket !B \rrbracket = !\llbracket B \rrbracket$ and then $\epsilon \mapsto \llbracket \partial_\beta \alpha \rrbracket$.

11.3.1 Encoding the product rule as a cut-elimination rule

We have shown how to define the derivative in terms of the linear semantics. It turns our there is a purely syntactic definition of the derivative in linear logic. It consists of adding codereliction and cocontraction rules along with corresponding cut-elimination rules. Then the derivative is

$$\dfrac{\dfrac{\overline{\quad}\,\pi}{!A \vdash B}}{!A, A \vdash A}\,\mathrm{DIFF} \qquad \text{is defined to be} \qquad \dfrac{\dfrac{\dfrac{\overline{\quad}\,\pi}{!A \vdash B}\,\mathrm{COCTR}}{!A, !A \vdash A}\,\mathrm{CODER}}{!A, A \vdash B}$$

Then the product rule is encoded in the following cut-elimination rule

$$\dfrac{\dfrac{\overline{\quad}}{!A, A \vdash B}\,\partial_A \qquad \dfrac{\dfrac{\overline{\quad}\,\pi}{!A, !A \vdash B}\,\mathrm{CTR}}{!A \vdash B}}{!A, A \vdash B}$$

reduces via cut-elimination to

$$\dfrac{\dfrac{\overline{\quad}}{!A, A \vdash !A}\,\partial_A \qquad \dfrac{\overline{\quad}}{!A, !A \vdash B}\,\pi}{\dfrac{!A, !A, A \vdash B}{!A, A \vdash B}\,\mathrm{CTR}}\,\mathrm{CUT} \quad + \quad \dfrac{\dfrac{\overline{\quad}}{!A, A \vdash !A}\,\partial_A \qquad \dfrac{\dfrac{\overline{\quad}\,\pi}{!A, !A \vdash B}\,\mathrm{EXC}}{!A, !A \vdash B}\,\mathrm{CUT}}{\dfrac{!A, !A, A \vdash B}{!A, A \vdash B}\,\mathrm{CTR}}$$

11.4 Example

11.5 Probabilistic Interpretation

We are going to think of $!V$ as the space of probability distributions on V. Think of $|\varnothing\rangle_v$ as the detla distribution at $v \in V$ and $|v_1, \ldots, v_n\rangle_v$ as the weak derivatives

$$\partial_{v_1} \cdots \partial_{v_n} v$$

meaning they integrate against a function $f : V \to k$ to give

$$(-1)^n \partial_{v_1} \cdots \partial_{v_n} f(0)$$

We think of the lifting

$$!\llbracket \psi \rrbracket : !\llbracket A \rrbracket \to !\llbracket B \rrbracket$$

as the natural extension of a measurable map to the space of distributions via pushforward measure.

Therefore, if we understand what the underlying spaces $\llbracket A \rrbracket$ and $\llbracket B \rrbracket$ represent for a given program, we can understand the derivative as an action on probability distributions (with finite support) over those spaces. Indeed we see that,

$$\partial_{\llbracket \beta \rrbracket} \llbracket \psi \rrbracket (\llbracket \alpha \rrbracket) = \llbracket \psi \rrbracket_* \nabla_{\llbracket \beta \rrbracket} \delta_{\llbracket \alpha \rrbracket}$$

Given $v_1, \ldots, v_n \in V$ and real numbers $0 \leq a_i \leq 1$ there are two encodings of their distributions

$$\sum_i a_i \, |\varnothing\rangle_{v_i} \qquad |\varnothing\rangle_{\sum a_i v_i}$$

the former we call the *standard encoding* and the later the *naive encoding*.