A CATEGORICAL APPROACH TO PROOFS-AS-PROGRAMS

CARSEN BERGER

Abstract. The correspondence between proofs in intuitionistic logic, programs in the λ-calculus, and cartesian closed categories is presented from a category-theoretic perspective. The main result of the paper is a proof that the category of positive implicational propositional theories is equivalent to a full subcategory of the category of λ-theories. This latter is then shown to be equivalent to the category of small cartesian closed categories, thus giving a categorical framework for the three-way correspondence between proofs, programs, and ccc's.

Contents

1. Introduction 1
2. Positive intuitionist propositional logic 2
3. λ-calculus 4
4. Curry-Howard, classically 9
5. Curry-Howard, categorically 10
6. Cartesian closed categories 15
7. Curry-Howard-Lambek, categorically 18
Acknowledgments 19
References 19

1. Introduction

It has been known for some time that there is a connection between propositional and combinatory logic. Haskell Curry remarked in the 1950s on a striking resemblance between the axioms of Hilbert-style deduction and the basic combinators of Curry's combinatory logic. To understand the full weight of Curry's observation it should be remembered that his aim was to set up combintory logic as a foundation for all of mathematics. Curry's combinatory system is equivalent to Church's λ-calculus (see Lambek [7] for a proof), which was designed to provide a model of computation. The well-known Church-Turing thesis states that any computable function is definable in the λ-calculus. I shall not pursue this tangent further, but hope merely to convey that the λ-calculus (equivalently, combinatory logic) is "sufficiently powerful" to be interesting. In this paper I eschew combinators in favor of λ-terms, but the reader seeking motivation should have always in mind the λ-calculus's close ties to combinatory logic and computability.

In 1969 W. A. Howard expanded Curry's observation to give a correspondence between proofs in natural deduction and constructions in the simply typed λ-calculus.

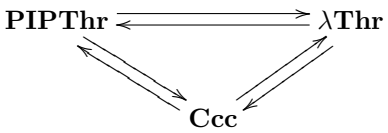
Date: September 1, 2010.

The λ-calculus corresponds to proofs in *intuitionistic* (or constructivist) logic, which in its most general form can be thought of as classical logic without the law of the exluded middle,

$$p \vee \neg p.$$

A proposition (or its negation) is not intuitionistically true until a proof has been constructed, regardless of whether its negation has been proven to be false. Since functions in the λ-calculus can be thought of as *programs*, which are inherently constructive, we should not expect to be able to arrive at a result by contradiction. There is no such thing as a "program by contradiction."[1]

In this paper I make two different uses of category theory in presenting the Curry-Howard correspondence between proofs and programs. My presentation of the isomorphism that lies at the heart of this correspondence uses categorical language; I show that the Curry-Howard isomorphism *is* an isomorphism of categories. Some excuse for this treatment of the material is to be found in the work of Lambek, which I also present. Lambek argued repeatedly that the λ-calculus—and hence also positive intuitionist propositional logic (PIPL)—is really just a cartesian closed category (ccc). This connection has deep semantical consequences (see, for example, Johnstone's section on categorical semantics [5, Section D4] and parts of Goldblatt [3]), but it is beyond the scope of this paper to delve into them. Taking after Awodey, I present Lambek's correspondence as an equivalence of categories—between the category of ccc's and the category of λ-theories. In other words, my goal will be to give an introduction to the following diagram.



There are many formal resemblances between these areas of proof theory, type theory, and category theory. My approach takes as its motto the wisdom of J. P. May, who once remarked that "the purpose of being categorical is to make that which is formal, formally formal."

The first half of this paper provides background material, with Sections 2 and 3 giving introductions to PIPL and the λ-calculus, respectively. In Section 4 I illustrate the Curry-Howard correspondence by adapting the original approach of Howard. The remainder of the paper assumes some familiarity with basic concepts of category theory. Section 5 contains the main result of the paper, a proof that the category of positive implicational propositional theories is equivalent to a full subcategory of the category of λ-theories. Section 6 presents some consequences of this by introducing cartesian closed categories and explaining their place in the picture, and Section 7 puts this explanation into categorical terms.

2. Positive intuitionist propositional logic

In this paper we will be considering propositional (zeroth-order) calculi, which are deductive systems defined over a set of *terms* or *formulae*. Generally we are

[1]There is actually a certain sense in which "program by contradiction" does make sense; it involves allowing nonsequential program execution. See Examples 5.2 and 5.7.

given a set of *atomic* or *prime* formulae, which intuitively can be thought of as possessing truth values (true or false). Here the prime formulae will be propositional variables, as well as a constant $\top$ for *truth*. It is also common to include a constant for falsity, and possibly non-logical constants, which can be thought of as *assumptions*. From these prime formulae we build up our set of terms by applying *logical connectives* (conjunction, disjunction, implication, negation) to existing terms. In general the truth values of these compound terms are determined by considering the connectives as functions acting on the truth values of their subterms.

In this section we are concerned with the syntax of the deductive system. That is, we are concerned with those formulae which are valid (derivable) in our logic, given a set of *axioms* and *rules of inference*. To this end we are concerned with the notion of *entailment*, and we write "$\alpha \vdash \beta$" for "α entails β," where α and β are formulae. We will be dealing with *sequents*, which are expressions of the form $\Gamma \vdash \beta$, where Γ is a finite, possibly empty, sequence of formulae (the assumptions) and β is a formula (the conclusion). Entailment can be thought of as implication in the metalogic. From this it is possible to arrive at the collection of formulae which become true if we assume the truth of some formula α; see Lambek [6], [7] for more about this process of adding indeterminates.

We will consider a fragment of intuitionist logic dealing only with implication ($\supset$), conjunction ($\wedge$), and truth ($\top$). We limit ourselves to this positive implicational fragment because of the ease with which it can be compared to the λ-calculus with explicit pairs and cartesian closed categories. It is possible to discover λ-calculus and category-theoretic analogues to an enriched intuitionist logic dealing also with negation, disjunction, falsity, and quantifiers (see, for example, Howard [4], Lambek [6], and Scott [9]) but the insight gained by considering such an extended system is marginal. The interested reader may wish to attempt the extension himself. In particular I will remark, however, that the corresponding categories will be not only cartesian, but bicartesian.

In any case we are now ready for our first definition. But first, a motivating word about the axioms and rules of inference. Dealing as we are with a positive and constructive logic, we want to allow *modus ponens* but have no place for such techniques as *modus tollens*, proof by contradiction, and proof by contrapositive. We also want introduction and elimination rules for conjunction and implication, as well as axioms for identity and truth and a rule for composition of entailment so that the logic resembles our intuitive notions.

Remark 2.1. Whenever an inductive definition is given, it is to be understood that the set defined is the smallest set satisfying the definition.

Definition 2.2. The *positive intuitionist propositional logic (PIPL)* is a 6-tuple $(\mathcal{V}, \mathcal{F}, \mathcal{P}, \supset, \wedge, \top)$ where:

(1) $\mathcal{V}$ is a countable set of propositional variables $v_0, v_1, v_2, \ldots$
(2) $\mathcal{F}$ is a set of *formulae* defined inductively as follows:

> Each propositional variable is a formula;
>
> $\top$ is a formula;
>
> If α and β are formulae, so are $\alpha \wedge \beta$ and $\alpha \supset \beta$.

(3) $\mathcal{P}$ is a set of *proofs*. A proof of a sequent $\Gamma \vdash \beta$ is an expression of the form $P : \Gamma \vdash \beta$. The set of proofs is defined inductively as follows, where the

horizontal bars indicate that the proofs on the top give us the proof on the bottom:

(a) The following are proofs:

$$1 : \alpha \vdash \alpha,$$
$$0 : \alpha \vdash \top,$$
$$\pi_1 : (\alpha \wedge \beta) \vdash \alpha,$$
$$\pi_2 : (\alpha \wedge \beta) \vdash \beta,$$
$$\varepsilon : ((\alpha \supset \beta) \wedge \alpha) \vdash \beta;$$

(b)
$$\frac{P : \Gamma \vdash \alpha \qquad Q : \alpha \vdash \beta}{(QP) : \Gamma \vdash \beta};$$

(c)
$$\frac{P : \Gamma \vdash \alpha \qquad Q : \Gamma \vdash \beta}{\langle P, Q\rangle : \Gamma \vdash \alpha \wedge \beta};$$

(d)
$$\frac{P : \Gamma, \alpha \vdash \beta}{\lambda P : \Gamma \vdash (\alpha \supset \beta)}.$$

Remark 2.3. (a) gives the axioms of our deductive system, while (b), (c), and (d) give the rules of inference. The symbols $1, 0, \ldots$ in (a) really denote *classes* of proofs, and should really be written with subscripts $1_\alpha, 0_\alpha, \ldots$ as appropriate, but I omit these for the sake of notational clarity when confusion is not likely to arise.

We say that a sequent $\Gamma \vdash \beta$ is a *theorem* if there exists a proof $P : \Gamma \vdash \beta$.

Example 2.4. The commutative law

$$\langle \pi_2, \pi_1\rangle : \alpha \wedge \beta \vdash \beta \wedge \alpha$$

is illustrated thus:

$$\frac{\pi_2 : \alpha \wedge \beta \vdash \beta \qquad \pi_1 : \alpha \wedge \beta \vdash \alpha}{\langle \pi_2, \pi_1\rangle : \alpha \wedge \beta \vdash \beta \wedge \alpha}.$$

3. λ-calculus

The λ-calculus is a model of computation. More explicitly, it is a theory of functions. Functions are of course familiar to us from set theory, where they are defined as sets of ordered pairs, associating inputs to their outputs. Functions in the λ-calculus are different in that the process by which an input is transformed into an output is of primary concern. It is this property of the λ-calculus that makes it a suitable model for computation. It can be thought of as a primitive (albeit highly inefficient) programming language, where programs are represented as functions. Indeed, real-world functional programming languges such as LISP have been based on it.

There are two fundamental operations associated with functions which we would like to capture in our theory. The first is *function application*, the operation of applying a function to an input term to produce an output term. If we think of a function as an infinite entity (recall Hilbert and his function spaces), function application is an instantiation of the function, a projection down into the realm of the fixed and finite. The second operation is *function abstraction*, which can

be thought of as inverse to application. Starting from a fixed term, we create an infinite entity which, when instantiated (applied to an input), always gives back our original term. Thus 5 is abstracted to $f(x) = 5$.

Note one other thing about that example: we have given our function a *name*, namely, f. We could also have written $x \mapsto 5$ for the unnamed function which maps x to 5. In the λ-calculus all functions are unnamed, and we express them using λ-notation. Thus we would write $\lambda x.5$ for the function which takes x to 5, or $\lambda n.n + 1$ for the function which adds 1 to its input.

The λ-calculus, as a language, is made up of terms. In the calculus's most basic form, a term t is either a variable x, a concatenation of two terms tt, or a λ-abstraction of a term, $\lambda x.t$. The syntax is as follows, where the vertical bars delimit rules of construction:

$$t := x \mid tt \mid \lambda x.t.$$

The semantics consists of one rule for application, called β-reduction for historical reasons:

$$(\lambda x.t)u \to_\beta t[x := u].$$

The expression on the right of the arrow is the notation for the term obtained by substituting u for all free occurences of the variable x in t.

Exercise 3.1. Define the natural numbers in this language. Hint: 2 is $\lambda f.\lambda x.f(fx)$.

Exercise 3.2. Define addition.

Definition 3.3. A variable x occurs *free* in a λ-term t if x is not in the scope of some λx; x occurs *bound* otherwise.

Example 3.4. A variable x may be free in t, but it is bound in $\lambda x.t$. The x in $\lambda x.xyz$ is bound, while the y and z are free. Both x and y are bound in $\lambda y.\lambda x.xyz$, but the z is still free.

Remark 3.5. This is similar to bound variables in expressions like $\forall x \ldots$ in predicate logic and $\int_0^1 \ldots \mathrm{d}x$ in calculus.

Unfortunately we soon run into trouble with terms such as $\lambda x.xx$. If f is a λ-term, let $b = \lambda x.f(xx)$ and put $a = bb$. Then $fa = f(bb) = bb = a$, which tells us that every term f has a fixed point. This becomes a problem when f is negation, since we cannot have $\neg a = a$ if a is a proposition. In essence, we have fallen into Russell's paradox.

And so we come through necessity to the *typed* λ-calculus, which avoids this issue. We will define types in a moment, but think of them as classes or sorts or any other such mechanism historically employed by the set-theorists to escape from Russell's trap. Let x have type A. We write $x : A$, or sometimes $x \in A$. The latter notation gives a neat set-theoretic interpretation, where the epsilon means inclusion as always. If $t : B$, we say that $\lambda x.t$ has type $A \to B$ and write $(\lambda x.t) : A \to B$. We sometimes write $\lambda x : A.t$ if we wish to emphasize the type of the argument. Of course, $A \to B$ can be given the natural set-theoretic interpretation as the function space of all set functions from A to B. So abusing notation it also makes sense to write $(\lambda x.t) \in (A \to B)$.

We now impose additional constraints on our syntax: it only makes sense to write fa if $f : A \to B$ and $a : A$. (So $fa : B$.) Expressions like $\lambda x.xx$ are no longer

definable; a respect for cardinality forbids sets isomorphic to their own function spaces.[2]

This is all the mechanics we need for a pure theory of (typed) functions, but there are natural extensions we can make. Our set-theoretic interpretation of types offers some suggestions. Consider the cartesian product of two types, $A \times B$. This is the *product type* whose objects are ordered pairs of the form $\langle t, u\rangle$ with $t : A$ and $u : B$. This allows us to define n-tuples in terms of pairs, a construction which will be familiar to those who have programmed in languages such as LISP. Having lists of objects allows our language to operate on arbitrarily large (finite) collections of data.

The introduction of types interacts in an important way with variable binding. Note that we can bind a free variable manually, as it were. An occurence of a variable is free only if we are free to give it whatever value we like, independent of the term it occurs in. An occurence of a variable is bound if its value is determined by an occurence of that variable in a higher scope. Such a higher scope is called a *context*, or an *environment*. We can bind a variable within the syntax of the λ-calculus itself by prefixing λx to a term with x free, but we could also bind variables on the metalinguistic level by specifying a context containing bindings for the free variables and interpreting our term in that context.

Definition 3.6. A *typing context*, or simply *context*, is a finite (possibly empty) sequence of distinct variables along with their types. The order of variables in the sequence matters, so *par abus de notation* we write $\Gamma = \langle x_1 : A_1, \ldots, x_n : A_n\rangle$. I will frequently omit the chevrons when writing the sequence, and will write $\Gamma, x : A$ for the context $\langle x_1 : A_1, \ldots, x_n : A_n, x : A\rangle$.

Remark 3.7. The chevron notation suggests a connection to the ordered pairs of our product type, and indeed $\langle x_1 : A_1, \ldots, x_n : A_n\rangle$ can be defined as an object of the n-fold product type $A_1 \times \ldots \times A_n$, which we get from iterating binary products in the obvious way. Thus the comma in shorthand expressions like $\Gamma, x : A$ should be understood as possessing well-defined pairing properties.

Is it possible to define the empty product type, whose object is the empty list $\langle\rangle$? We will return to this question in Section 6, at which point any handwaving will be cleared up.

Definition 3.8. A *typing judgment* is an interpretation of a term in a context. We write $\Gamma \vdash t : A$ and read "in context Γ the term t has type A."

Remark 3.9. We will always speak of typing judgments in which (at least) all free variables of t occur in Γ. Such a context gives a *closure* of t. We also use the word *closure* to refer to the closed term $\lambda\vec{x}.t = \lambda x_1 \ldots \lambda x_n.t$, where $\vec{x} = x_1, \ldots, x_n$ are all the free variables in t. Think of a closure of a term as filling in all the holes left by its free variables.

In our language the type of a variable will be determined by its context. Thus we might have $x : A$ in one context and $x : B$ in another. This allows us to construct

[2]However, we *can* define types isomorphic to their own function spaces if we leave the category **Set**. See Scott's seminal paper [8] for a topologically-oriented construction of such a reflexive type in the category of continuous lattices (as well as a proof that every topological space can be embedded in such a lattice). This provides a model for the untyped λ-calculus. It is a short step from there to seeing that the untyped λ-calculus can really be considered a typed theory with a single type, as Scott argues in [9].

polymorphic functions—that is, functions which can operate on different types in different contexts.

Example 3.10. Consider the term $t = \lambda x\colon A.\langle x, y\rangle$. Since y is free its type is not determined, so we might write $t\colon A \to A \times \mu$, where μ is a *type variable* ranging over the set of types.

Thus t will take on different types in different contexts; for example,

$$\Gamma, y\colon A \vdash t\colon A \to A \times A \quad \text{and} \quad \Gamma, y\colon B \vdash t\colon A \to A \times B.$$

The last element we need in our definition of the λ-calculus is a set of axioms and rules to express the semantics of the language. There are no connectives or quantifiers in the λ-calculus; it is an equational theory. Syntactic equality, however, is too strong to be very useful; we would like a weaker notion of equivalence. The basic notion of equivalence between terms is that of *convertibility*, which will be defined by the axioms below.

I will use horizontal bars to indicate that we can get one typing judgment from another.

Definition 3.11. The *typed λ-calculus with explicit pairs* is a 4-tuple $(\mathrm{Typ}, \mathcal{V}, \Lambda, \mathcal{A})$ where:

(1) Typ is a set of *types* defined inductively as follows:
   (a) 1 is a type, called the *unit type*;
   (b) If A and B are types, so are $A \times B$ and $A \to B$ (product and function types).

(2) $\mathcal{V}$ is a countable set of *variables* $x_0, x_1, x_2, \ldots$

(3) Λ is a set of *terms*, together with their types. In other words, we want to give a typing judgment for each element of Λ; thus we can consider Λ to be the set of *terms in context* (where the context is possibly empty or the term itself). Λ is defined inductively as follows.
   (a) The type of a variable is determined by its context:
   $$\Gamma, x\colon A \vdash x\colon A.$$
   (b) There is a unit constant:
   $$\Gamma \vdash *\colon 1.$$
   (c) Binary pairs and projections:
      (i)
      $$\frac{\Gamma \vdash t\colon A \qquad \Gamma \vdash u\colon B}{\Gamma \vdash \langle t, u\rangle\colon A \times B};$$
      (ii)
      $$\Gamma, t\colon A \times B \vdash \pi_1(t)\colon A, \qquad \Gamma, t\colon A \times B \vdash \pi_2(t)\colon B.$$
   (d) Function application and abstraction:
      (i)
      $$\Gamma, f\colon A \to B, a\colon A \vdash fa\colon B;$$
      (ii)
      $$\frac{\Gamma, x\colon A \vdash t\colon B}{\Gamma \vdash (\lambda x.t)\colon A \to B}.$$

(4) $\mathcal{A}$ is a set of *axioms* governing equivalence of terms, and is defined as follows:

(a) (i)
$$\Gamma \vdash t = t : A,$$
(ii)
$$\frac{\Gamma \vdash t = u : A}{\Gamma \vdash u = t : A},$$
(iii)
$$\frac{\Gamma \vdash t = u : A \qquad \Gamma \vdash u = v : A}{\Gamma \vdash t = v : A},$$
(iv)
$$\frac{\Gamma \vdash t = u : A}{\Gamma, x : B \vdash t = u : A}.$$
(b) Unit type:
$$\Gamma \vdash t = * : 1.$$
(c) Product types:
(i)
$$\frac{\Gamma \vdash u = v : A \qquad \Gamma \vdash s = t : B}{\Gamma \vdash \langle u, s\rangle = \langle v, t\rangle : A \times B};$$
(ii)
$$\frac{\Gamma \vdash s = t : A \times B}{\Gamma \vdash \pi_1(s) = \pi_1(t) : A}, \qquad \frac{\Gamma \vdash s = t : A \times B}{\Gamma \vdash \pi_2(s) = \pi_2(t) : B};$$
(iii)
$$\Gamma \vdash t = \langle \pi_1(t), \pi_2(t)\rangle : A \times B;$$
(iv)
$$\Gamma \vdash \pi_1(\langle u, t\rangle) = u : A, \qquad \Gamma \vdash \pi_2(\langle u, t\rangle) = t : B;$$
(d) Function types:
(i)
$$\frac{\Gamma \vdash s = t : A \to B \qquad \Gamma \vdash u = v : A}{\Gamma \vdash su = tv : B};$$
(ii) Rule β:
$$\Gamma \vdash (\lambda x : A.t)u = t[x := u] : B, \text{ where } t : B;$$
(iii) Rule ξ:
$$\frac{\Gamma, x : A \vdash t = u : B}{\Gamma \vdash \lambda x.t = \lambda x.u : A \to B};$$
(iv) Rule η:
$$\Gamma \vdash \lambda x.tx = t : A \to B \text{ if } x \text{ is not a free variable of } t.$$

Notation 3.12. I will refer to this as simply *the* λ*-calculus*. It is also occasionally called the $\lambda\beta\eta\pi$-calculus to emphasize the presence of the rules β and η, and the π for pairs and projections. (The names β, ξ and η are historical.) The reader will forgive me if I assume the $\beta\eta\pi$ to be understood and speak no more of it.

4. Curry-Howard, classically

The attentive reader may have noticed some similarities between the definition of the λ-calculus and the definition of PIPL. In this section I clarify these similarities by presenting a theorem due to Howard [4] (extended to match the definitions in this paper). We will consider a variant of PIPL, call it PIPL$'$, whose only prime formula is $\top$. That is, we identify the propositional variables with truth. In the next section we will see what happens when we consider more complicated systems.

Construction 4.1. Define an equivalence relation $\sim$ on $\mathcal{F}$, the set of formulae in PIPL, as follows:

$$\begin{aligned}
&v_i \sim \top \text{ for all } v_i \in \mathcal{V},\\
&(\alpha \wedge \beta) \sim (\alpha' \wedge \beta') \text{ if } \alpha \sim \alpha' \text{ and } \beta \sim \beta',\\
&(\alpha \supset \beta) \sim (\alpha' \supset \beta') \text{ if } \alpha \sim \alpha' \text{ and } \beta \sim \beta'.
\end{aligned}$$

Let $\mathcal{F}' = \mathcal{F}/\sim$, and let PIPL$'$ be defined over $\mathcal{F}'$ as in Definition 2.2.

Let $\phi : \mathcal{F}' \to \mathrm{Typ}$ be the map from formulae to types defined as follows:

$$\begin{aligned}
&\phi([\top]) = 1,\\
&\phi([\alpha \wedge \beta]) = \phi([\alpha]) \times \phi([\beta]),\\
&\phi([\alpha \supset \beta]) = \phi([\alpha]) \to \phi([\beta]).
\end{aligned}$$

Notation 4.2. Clearly ϕ gives a bijection between $\mathcal{F}'$ and Typ. I will use small Greek letters $\alpha, \beta, \gamma, \ldots$ to refer to (equivalence classes of) formulae and large Roman letters $A, B, C, \ldots$ to refer to types, writing A rather than $\phi(\alpha)$.

Notation 4.3. In the following discussion I will use Γ to refer to sequences of PIPL$'$ formulae and Δ to refer to closed λ-contexts (that is, contexts which contain bindings for all free variables in the term in question).

Definition 4.4. A *construction* of a sequent $\Gamma \vdash \beta$ in PIPL$'$ is a λ-term in context $\Delta \vdash t : B$ such that for every free variable $x : A$ occuring in $t : B$ (which becomes bound in Δ) there is a corresponding occurence of α in Γ (it being understood that the existence of k distinct free variables of the same type A in $t : B$ is reflected by at least k occurences of α in Γ).

Remark 4.5. By a *free variable* here I mean a variable which is free in $t : B$ but becomes bound in $\Delta \vdash t : B$. The context Δ is required, however, in order to give the free variable a type.

Examples 4.6. (1) $x : A \vdash x : A$ is a construction of $\alpha \vdash \alpha$ as well as of $\alpha, \beta \vdash \alpha$.
(2) $\vdash (\lambda x.\lambda y.x) : A \to (B \to A)$, the constant combinator **K** of combinatory logic, is a construction of $\vdash \alpha \supset (\beta \supset \alpha)$, the first axiom of Hilbert-style deduction.

Theorem 4.7 (Howard)**.** *Given any proof of $\Gamma \vdash \beta$ in PIPL$'$ we can find a construction of $\Gamma \vdash \beta$, and conversely.*

Proof. Define a map $\phi : \mathcal{P}' \to \Lambda$ from proofs to constructions as follows, inducting on the structure of proof.

$$\phi(\alpha \vdash \alpha) = x : A \vdash x : A$$
$$\phi(\alpha \vdash \top) = x : A \vdash * : 1$$
$$\phi(\alpha \wedge \beta \vdash \alpha) = t : A \times B \vdash \pi_1(t) : A$$
$$\phi(\alpha \wedge \beta \vdash \beta) = t : A \times B \vdash \pi_2(t) : B$$
$$\phi((\alpha \supset \beta) \wedge \alpha \vdash \beta) = f : A \to B, a : A \vdash fa : B$$
$$\frac{\phi(P) = \Delta \vdash t : A \qquad \phi(Q) = x : A \vdash u : B}{\phi(QP) = \Delta \vdash u[x := t] : B}$$
$$\frac{\phi(P) = \Delta \vdash t : A \qquad \phi(Q) = \Delta \vdash u : B}{\phi(\langle P, Q \rangle) = \Delta \vdash \langle t, u \rangle : A \times B}$$
$$\frac{\phi(P) = \Delta, x : A \vdash t : B}{\phi(\lambda P) = \Delta \vdash (\lambda x.t) : A \to B}$$

Thus every proof has a construction. The converse is proved similarly, although care must be taken to ensure that all variables in the context are accounted for. □

Remark 4.8. A λ-term with no free variables is called a *combinator*. Clearly combinators correspond to axioms in our deductive system.

Observe that a formula β is provable from a set of assumptions Γ if and only if there exists a term t of type B in the context Δ which contains variable bindings $x : A$ for all $\alpha \in \Gamma$. In other words,

$$\alpha_1, \ldots, \alpha_n \vdash \beta \iff x_1 : A_1, \ldots, x_n : A_n \vdash t : B.$$

Thus a formula is provable precisely when its corresponding type is inhabited. This gives a feel for just how constructive intuitionist logic really is; we have actually constructed an object corresponding to each true statement in PIPL. Furthermore, we have seen that every object that can be so constructed (that is, every computable function, if we recall the Church-Turing thesis) has a canonical theorem in PIPL.

5. Curry-Howard, categorically

We now go a step further and show that not only is PIPL equivalent to the λ-calculus, but the category of all deductive theories that "look like" PIPL is equivalent to a subcategory of the category of all combinatory theories that "look like" the λ-calculus. The motivation for this excursion will be become clear when we encounter cartesian closed categories; there is *a* ccc (up to equivalence) to which the λ-calculus is equivalent, but there are many other ccc's found in the wild. It is a natural step to identify an internal language for each of them, and conversely, to give a categorical model for each theory. We will take up this work in Section 6, but first we must elucidate the relationship between deductive and combinatory theories. We begin by constructing the category of positive implicational propositional theories.

Definition 5.1. A *positive implicational propositional theory (PIP theory)* $\mathbb{T}$ is a 7-tuple $(\mathcal{V}, \mathcal{F}, \mathcal{A}, \mathcal{P}, \supset, \wedge, \top)$ where:

(1) $\mathcal{V}$ is a (possibly empty) set of *atomic formulae*, which we may as well take to be propositional variables;

(2) $\mathcal{F}$ is the smallest set satisfying Definition 2.2 such that $\mathcal{V} \subset \mathcal{F}$;
(3) $\mathcal{A}$ is a (possibly empty) set of *non-logical axioms*—that is, a collection of sequents;
(4) $\mathcal{P}$ is the smallest set satisfying Definition 2.2 such that $\mathcal{A} \subset \mathcal{P}$;
(5) and $\supset$, $\wedge$ and $\top$ are as in Definition 2.2.

Note that a PIP theory is uniquely determined by only two pieces of information: its set $\mathcal{V}$ of atomic formulae and its set $\mathcal{A}$ of axioms.

Examples 5.2. (1) If $\mathcal{A} = \emptyset$ then $\mathbb{T}$ is just PIPL over $\mathcal{V}_\mathbb{T}$.
(2) If $\mathcal{A} = \{\top \vdash \alpha\}$, where α is a propositional variable, then $\mathbb{T}$ is the theory obtained by adjoining the assumption α to PIPL.
(3) If $\mathcal{A} = \{(\alpha \supset \beta) \supset \alpha \vdash \alpha\}$, then $\mathbb{T}$ is the positive implicational fragment of classical logic over $\mathcal{V}_\mathbb{T}$. This axiom is Peirce's law, which is equivalent to the law of the excluded middle; it is not provable in PIPL alone.

We now need to define morphisms between PIP theories. I will first present the definition and then discuss why it is the right definition.

Definition 5.3. A *p-translation* (or simply *translation*) $\tau : \mathbb{T} \to \mathbb{U}$, where $\mathbb{T}$ and $\mathbb{U}$ are PIP theories, is a map satisfying the following conditions:
(1) Each propositional variable in $\mathbb{T}$ maps to a propositional variable in $\mathbb{U}$, and $\tau : \mathcal{V}_\mathbb{T} \to \mathcal{V}_\mathbb{U}$ is injective;
(2)

$$\begin{aligned} \tau\top &= \top, \\ \tau(\alpha \wedge \beta) &= \tau\alpha \wedge \tau\beta, \\ \tau(\alpha \supset \beta) &= \tau\alpha \supset \tau\beta; \end{aligned}$$

(3) Each sequent $\alpha_1, \ldots, \alpha_n \vdash \beta$ maps to the sequent $\tau\alpha_1, \ldots, \tau\alpha_n \vdash \tau\beta$;
(4) Each non-logical axiom $(\Gamma \vdash \alpha) \in \mathcal{A}_\mathbb{T}$ maps to a non-logical axiom $(\tau\Gamma \vdash \tau\alpha) \in \mathcal{A}_\mathbb{U}$;
(5) $\tau : \mathcal{P}_\mathbb{T} \to \mathcal{P}_\mathbb{U}$ is defined inductively on the structure of proofs in the canonical manner.

It should be clear that if $\Gamma \vdash \beta$ is a theorem in $\mathbb{T}$, then $\tau\Gamma \vdash \tau\beta$ is a theorem in $\mathbb{U}$; translations preserve provability, and all structure associated with it. Thus a translation can be considered an *embedding* of one theory into another.

Definition 5.4. PIPThr is the category whose objects are PIP theories and whose morphisms are p-translations.

It is easy to check that **PIPThr** is indeed a category. Translations compose as functions, so composition is associative. The identity translation which takes every variable, formula, sequent, axiom and proof to itself gives the identity morphism.

Remark 5.5. By the way, translations (embeddings) give a natural ordering on theories. Define a relation $\prec$ over PIP theories by saying $\mathbb{T} \prec \mathbb{U}$ if there exists a translation $\tau : \mathbb{T} \to \mathbb{U}$. **PIPThr** is not a pre-order category however, because there may be (and usually are) multiple ways to translate $\mathbb{T}$ into $\mathbb{U}$. Each such translation is a *model* of $\mathbb{T}$ in $\mathbb{U}$.

We now turn our attention to λ-theories.

Definition 5.6. A *typed λ-theory with explicit pairs (λ-theory)* $\mathbb{L}$ is a 7-tuple $(\mathcal{T}, \mathrm{Typ}, \mathcal{V}, \mathcal{K}, \Lambda, \mathcal{E}, \mathcal{A})$ where:

(1) $\mathcal{T}$ is a (possibly empty) set of *basic types*;
(2) Typ is the smallest set satisfying Definition 3.11 such that $\mathcal{T} \subset \mathrm{Typ}$;
(3) $\mathcal{V}$ is a countable set of variables as in Definition 3.11;
(4) $\mathcal{K}$ is a (possibly empty) set of *basic constants* (together with their types of course);
(5) Λ is the smallest set satisfying Definition 3.11 such that $\mathcal{K} \subset \Lambda$;
(6) $\mathcal{E}$ is a (possibly empty) set of *non-logical axioms*—that is, equations between terms;
(7) $\mathcal{A}$ is the smallest set satisfying Definition 3.11 such that $\mathcal{E} \subset \mathcal{A}$.

Note that a λ-theory is uniquely determined by its sets $\mathcal{T}$, $\mathcal{K}$ and $\mathcal{E}$.

Examples 5.7. (1) If $\mathcal{T} = \mathcal{K} = \mathcal{E} = \emptyset$ then $\mathbb{L}$ is just the λ-calculus.
(2) Let $\mathcal{K} = \{f : (A \to B) \to A \vdash t : A \mid A, B \in \mathrm{Typ}\}$. $((A \to B) \to A) \to A$ is the type of a control flow operator known as *call-with-current-continuation* (call/cc), which allows nonsequential program execution. It gives a construction for Peirce's law, so $\mathbb{L}$ in this example is the λ-theory corresponding to classical logic.

Definition 5.8. A *λ-translation* $\tau : \mathbb{L} \to \mathbb{M}$, where $\mathbb{L}$ and $\mathbb{M}$ are λ-theories, is a map satisfying the following conditions:

(1) Each basic type of $\mathbb{L}$ maps to a basic type of $\mathbb{M}$, and τ is injective on types;
(2)

$$\begin{aligned} \tau 1 &= 1, \\ \tau(A \times B) &= \tau A \times \tau B, \\ \tau(A \to B) &= \tau A \to \tau B; \end{aligned}$$

(3) Each variable of $\mathbb{L}$ maps to a variable of $\mathbb{M}$, and τ is injective on variables;
(4) A term in context $x_1 : A_1, \ldots, x_n : A_n \vdash t : B$ maps to a term in context $\tau x_1 : \tau A_1, \ldots, \tau x_n : \tau A_n \vdash \tau t : \tau B$;
(5) Each basic constant $\Gamma \vdash t : B$ maps to a basic constant $\tau\Gamma \vdash \tau t : \tau B$, and τ is injective on constants;
(6) $\tau * = *$, and the rest of $\Lambda_{\mathbb{L}}$ is mapped injectively into $\Lambda_{\mathbb{M}}$ by inducting on the structure of terms;
(7) If $\Gamma \vdash t = u : A$ is an equation in $\mathbb{L}$, then $\tau\Gamma \vdash \tau t = \tau u : \tau A$ is an equation in $\mathbb{M}$.

Our comments about p-translations also apply to λ-translations. All equations provable in $\mathbb{L}$ are translated into valid equations in $\mathbb{M}$. A λ-translation is a model of $\mathbb{L}$ in $\mathbb{M}$.

Definition 5.9. **λThr** is the category whose objects are λ-theories and whose morphisms are λ-translations.

The verification that **λThr** is a category is just the same as for **PIPThr**.

Before I present the main result of this section, we must discuss what constitutes an isomorphism of theories. We have observed that a PIP theory (resp. λ-theory) is uniquely deterimined by its sets $\mathcal{V}$ and $\mathcal{A}$ ($\mathcal{T}$, $\mathcal{K}$ and $\mathcal{E}$). *Ceteris paribus*, the theory whose set of atomic formulae is $\{\alpha_1, \alpha_2\}$ is distinct from the theory whose

set of atomic formulae is {"$2 + 2 = 5$", "The Eiffel Tower is tall."}; however, the sets $\mathcal{V}$ of these two theories are isomorphic—they each have the same number of elements. In some situations we may wish to consider generically "the theory with two prime formulae (and no non-logical axioms)," without regard for what those formulae are. This is useful in investigating categorical and structural properties of theories.

Proposition 5.10. *Two PIP theories (resp. λ-theories) are isomorphic objects of* **PIPThr** *(λ***Thr***) if and only if their sets $\mathcal{V}$ and $\mathcal{A}$ ($\mathcal{T}$, $\mathcal{K}$ and $\mathcal{E}$) are isomorphic as sets.*

Proof. Two theories $\mathbb{T}$ and $\mathbb{U}$ are isomorphic only if there exists a pair of iso arrows $\tau : \mathbb{T} \to \mathbb{U}$ and $\tau^{-1} : \mathbb{U} \to \mathbb{T}$. Since translations are injective on prime formulae (basic types), the sets $\mathcal{V}_\mathbb{T}$ and $\mathcal{V}_\mathbb{U}$ ($\mathcal{T}_\mathbb{T}$ and $\mathcal{T}_\mathbb{U}$) must have the same cardinality, and are therefore isomorphic. Similarly for $\mathcal{A}$ ($\mathcal{K}$ and $\mathcal{E}$).

In the other direction, if these sets are isomorphic then there exists a translation of $\mathbb{T}$ into $\mathbb{U}$ and a translation of $\mathbb{U}$ into $\mathbb{T}$. It is trivial to see that each $\tau : \mathbb{T} \to \mathbb{U}$ induces an inverse arrow, making it iso. □

Remark 5.11. In the notation of Remark 5.5, $\mathbb{T}$ is isomorphic to $\mathbb{U}$ if and only if $\mathbb{T} \prec \mathbb{U}$ and $\mathbb{U} \prec \mathbb{T}$.

A *skeleton* of a category is a full subcategory whose objects are representative elements of the isomorphism classes of the original category. In other words, in a skeleton *isomorphic* really does mean *is*.

Theorem 5.12. *Let* $\mathbf{P}$ *be a skeleton of* **PIPThr** *and let* $\mathbf{L}$ *be a skeleton of* λ**Thr**. *Then* $\mathbf{P}$ *is fully embeddable into* $\mathbf{L}$.

Proof. Let $\mathbf{S}$ be the full subcategory of $\mathbf{L}$ whose objects are the λ-theories with no non-logical axioms and which contain at most one basic constant of a given type; that is, $\mathcal{E} = \emptyset$ and $(\Gamma \vdash t : B), (\Gamma \vdash u : B) \in \mathcal{K} \implies t = u$. We will construct a functor $F : \mathbf{P} \to \mathbf{S} \subset \mathbf{L}$ and show that it is bijective on objects and on morphism sets, thus giving an isomorphism of categories.

Let $\mathbb{T} \in \mathbf{P}$ be a PIP theory. We define its image $F\mathbb{T}$ to be the smallest λ-theory satisfying the following conditions:

(1) For each atomic formula in $\mathbb{T}$ there is a basic type in $F\mathbb{T}$, and F is injective on formulae;

(2)
$$\begin{aligned} F\top &= 1, \\ F(\alpha \wedge \beta) &= F\alpha \times F\beta, \\ F(\alpha \supset \beta) &= F\alpha \to F\beta; \end{aligned}$$

(3) For each non-logical PIP axiom $\alpha_1, \ldots, \alpha_n \vdash \beta$ there is a basic constant $x_1 : F\alpha_1, \ldots, x_n : F\alpha_n \vdash t : F\beta$, and F is injective on axioms;

(4) $F : \mathcal{P}_\mathbb{T} \to \Lambda_{F\mathbb{T}}$ is defined inductively on the structure of proofs as in Theorem 4.7.

Now to show that F is injective on objects, suppose $F\mathbb{T} = F\mathbb{U}$. Then we get $\mathcal{T}_{F\mathbb{T}} = \mathcal{T}_{F\mathbb{U}}$ and $\mathcal{K}_{F\mathbb{T}} = \mathcal{K}_{F\mathbb{U}}$. By the requirement that $F\mathbb{T}$ be the *smallest* λ-theory satisfying conditions (1)-(4), and by the injectiveness requirement in conditions (1) and (3), we have established that the mappings of atomic formulae into basic types

and of non-logical axioms into basic constants are both surjective and injective. Thus, $\mathcal{V}_\mathbb{T} \cong \mathcal{V}_\mathbb{U}$ and $\mathcal{A}_\mathbb{T} \cong \mathcal{A}_\mathbb{U}$. But since we are in the skeleton these isomorphisms imply identities, and so $\mathbb{T} = \mathbb{U}$.

To show surjectivity, let $\mathbb{L} \in \mathbf{S}$. It is trivial to see that there is a $\mathbb{T} \in \mathbf{P}$ such that $F\mathcal{V}_\mathbb{T} = \mathcal{T}_\mathbb{L}$ and $F\mathcal{A}_\mathbb{T} = \mathcal{K}_\mathbb{L}$. Thus $F\mathbb{T} = \mathbb{L}$.

We define the action of F on morphisms as follows. Let $\tau : \mathbb{T} \to \mathbb{U}$ be a p-translation. If x is a formula or proof of $\mathbb{T}$, define $F\tau : F\mathbb{T} \to F\mathbb{U}$ by setting $F\tau(Fx) = F(\tau x)$. In other words, $F\tau$ is the unique morphism making the diagram

$$\begin{array}{ccc} \mathbb{T} & \xrightarrow{F} & F\mathbb{T} \\ {\scriptstyle\tau}\downarrow & & \downarrow{\scriptstyle F\tau} \\ \mathbb{U} & \xrightarrow{F} & F\mathbb{U} \end{array}$$

commute.

Let $\tau, \sigma : \mathbb{T} \to \mathbb{U}$ be p-translations. If $\tau \neq \sigma$, then there is some $x \in \mathbb{T}$ such that $\tau x \neq \sigma x$. By the bijectivity of F, this gives us $F(\tau x) \neq F(\sigma x)$. Thus $F\tau(Fx) \neq F\sigma(Fx)$, so $F\tau \neq F\sigma$. Thus F is injective on morphism sets.

Let $\sigma : F\mathbb{T} \to F\mathbb{U}$ be a λ-translation. By the bijectivity of F we can define a p-translation τ such that $\tau x = F^{-1}(\sigma(Fx))$. In other words, there is a unique τ making the diagram

$$\begin{array}{ccc} \mathbb{T} & \xrightarrow{F} & F\mathbb{T} \\ \downarrow{\scriptstyle\tau} & & {\scriptstyle\sigma}\downarrow \\ \mathbb{U} & \xleftarrow{F^{-1}} & F\mathbb{U} \end{array}$$

commute. So F is surjective on morphism sets.

Thus F gives an isomorphism of categories between $\mathbf{P}$ and $\mathbf{S}$, and is therefore a full embedding of $\mathbf{P}$ into $\mathbf{L}$. □

Proposition 5.13. PIPThr *is equivalent to a full subcategory of* λ**Thr**.

Proof. Two categories are equivalent if and only if their skeletons are isomorphic. □

It should be clear by now that a λ-theory contains more information than its corresponding PIP theory; a PIP theory determines the basic types and basic constants (up to inhabitance)[3] of a λ-theory, but says nothing about the latter's non-logical equational axioms. Up to isomorphism of theories, the picture is as follows. To borrow terminology from algebraic topology (which in turn borrowed terminology from botany), λ**Thr** can be regarded as a bundle with **PIPThr** serving as the base space. A given PIP theory $\mathbb{T}$ deterimines a set of λ-theories which share the same basic types and basic constants, up to inhabitance, but which differ in their non-logical axioms and in the number of basic constants of a given type. These λ-theories form a stalk, or fiber, over $\mathbb{T}$. Conversely, there is a forgetful functor—the submersion of the bundle—which takes a λ-theory to the PIP theory with the

[3]By *up to inhabitance* I mean that the axioms of a PIP theory determine whether or not there exists at least one basic constant of a given type in the corresponding λ-theory, but if a type is inhabited, they say nothing about how many basic constants there are of that type.

corresponding sets of atomic formulae and axioms, while forgetting about the equations of the λ-theory. Clearly each λ-theory maps to exactly one such PIP theory (up to isomorphism), and the mapping is not injective.

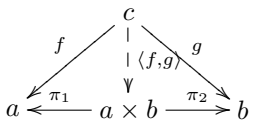
What is the point of all this? As we will see below, a λ-theory can be regarded as a cartesian closed category; its types will be objects and its terms in context will be arrows. A PIP theory can almost be regarded as a ccc, with formulae as objects; what will its arrows be? Not proofs, but equivalence classes of proofs. The equivalence relation is given to us for free when we consider proofs as terms in context. There are, of course, many different ways of taking equivalence classes of proofs of a PIP theory $\mathbb{T}$. Each λ-theory in the stalk over $\mathbb{T}$ is such a way; each germ in the stalk gives a different ccc.

6. Cartesian closed categories

We want to regard our theories as categories. Formulae (types) will be objects and equivalence classes of proofs (terms in context) will be morphisms. But due to the complicated definitions of our theories, the categories so given will possess rather specific structure. Let's jump right into it.

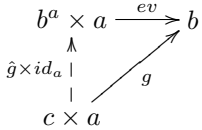
Definition 6.1. A *terminal object* in a category $\mathcal{C}$ is an object 1 such that for every object a there is exactly one morphism $*_a : a \to 1$.

Definition 6.2. A *product* of two objects a and b is an object $a \times b$ together with two morphisms $\pi_1 : a \times b \to a$ and $\pi_2 : a \times b \to b$ such that for any object c and any pair of morphisms $f : c \to a$ and $g : c \to b$ there is exactly one morphism $\langle f, g \rangle : c \to a \times b$ making the diagram



commute.

Definition 6.3. An *exponent* of two objects a and b is an object b^a together with a morphism $ev : b^a \times a \to b$, called an *evaluation arrow*, such that for any object c and any morphism $g : c \times a \to b$ there is exactly one morphism $\hat{g} : c \to b^a$ making the diagram



commute.

The morphisms g and $\hat{g}$ are called *exponential adjoints* of each other. Note that this establishes a bijection between morphism sets, $\mathcal{C}(c \times a, b) \cong \mathcal{C}(c, b^a)$.

Definition 6.4. A *cartesian closed category (ccc)* is a category with a terminal object, a product for any two objects, and an exponent for any two objects.

The 'cartesian' refers to products and the 'closed' refers to exponents. Equivalently we could define a ccc by requiring exponents and finitary (as opposed to binary) products. Note the lack of a terminal object in this definition. The reader

may wish to prove the equivalence of these definitions himself (it is not too hard, and quite enlightening); I will satisfy myself here by discussing just enough of it to give an answer to the question posed in Remark 3.7. I asked whether it was possible, in a λ-theory with primitive binary products, to define the empty product. The answer is no—unless we also have a primitive unit type.

Anyone who has programmed in LISP knows how to build a list (a term of a finite product type) out of pairs (terms of binary product types). Inductively, you set the first element of the pair to be the first element of the list, and the second element of the pair to be the rest of the list. The rest of the list is just another pair, whose first element is the next element in the list and whose second element is again the rest of the list. Of course well-founded inductions must have a base case, and so we require a term of the unit type to place in the second slot of the last pair of the list. This is sometimes called NULL or NIL; I have called it $*$. It is the empty list, and its type is the empty product. Without the unit type, then, we could not turn binary products into finite products.

Of course the other direction is trivial; finite products automatically give us the empty product, which gives us a unit type.

In any case, for a categorical interpretation of a theory to be useful it ought to satisfy the equations of the theory. This means that if $\Gamma \vdash t = u : A$ is an axiom in our theory, then u and t have the same interpretation in the model. This is why arrows will be equivalence classes of terms over the equivalence relation given by the axioms. We will construct a canonical model of a λ-theory called the *syntactic category*.

Definition 6.5. The *syntactic category* $\mathcal{S}(\mathbb{L})$ of a λ-theory $\mathbb{L}$ is the category whose objects are the types of $\mathbb{L}$ and whose morphisms $A \to B$ are terms in context $x : A \vdash t : B$, where two such terms $x : A \vdash t : B$ and $x : A \vdash u : B$ represent the same morphism if $x : A \vdash t = u : B$ is an equation in $\mathbb{L}$. Composition of terms $x : A \vdash t : B$ and $y : B \vdash u : C$ gives the term $x : A \vdash u[y := t] : C$. The identity morphism on A is given by $x : A \vdash x : A$.

Remark 6.6. We are here of course regarding a context as a term of the n-fold product type, as motivated by Remark 3.7.

Proposition 6.7. *If $\mathbb{L}$ is a λ-theory then its syntactic category $\mathcal{S}(\mathbb{L})$ is cartesian closed.*

Proof. The unit type 1 is the terminal object, with morphisms $*_A : A \to 1$ being given by terms $x : A \vdash * : 1$. Uniqueness is given by the axiom $\Gamma \vdash t = * : 1$.

The product of A and B is the product type $A \times B$ with the first and second projections being given by the terms $t : A \times B \vdash \pi_1(t) : A$ and $t : A \times B \vdash \pi_2(t) : B$. The exponent of A and B is the function type $A \to B$ with the evaluation arrow being given by the term $f : A \to B, a : A \vdash fa : B$.

The verification that this satisfies the definitions of products and exponentials is straightforward from the axioms of $\mathbb{L}$. □

Remark 6.8. The exponential adjoints here, of course, are the dual processes of function application and λ-abstraction. It is probable that Haskell Curry and Alonzo Church, who worked before the advent of category theory, would be both surprised and pleased to learn of this.

Remark 6.9. It is the lack of equations between proofs which prevents a similar construction of a syntactic category directly from a PIP theory from being cartesian closed. The logical equations in $\mathcal{A}_{\mathbb{L}}$ provide the smallest equivalence relation needed to make the syntactic category a ccc; the non-logical equations in $\mathcal{E}_{\mathbb{L}}$ determine the finer details of the category. Thus, each λ-theory in the stalk over a PIP theory $\mathbb{T}$ turns $\mathbb{T}$ into a different ccc.

The syntactic category allows us to regard models as functors.

Definition 6.10. A *model* M of a λ-theory $\mathbb{L}$ in a cartesian closed category $\mathcal{C}$ is a functor $M : \mathcal{S}(\mathbb{L}) \to \mathcal{C}$ that preserves finite products and exponentials.

We say that a model *satisfies* an equation $\Gamma \vdash t = u : A$ if the interpretations of u and t coincide:

$$M(\Gamma \vdash t : A) = M(\Gamma \vdash u : A).$$

As it so happens, the semantics of λ-theories in cartesian closed categories is both sound and complete.

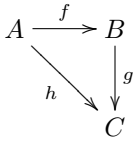
Proposition 6.11. *If $\mathbb{L}$ is a λ-theory, then $\mathbb{L}$ proves $\Gamma \vdash t = u : A$ if and only if every model of $\mathbb{L}$ satisfies $\Gamma \vdash t = u : A$.*

Proof. See Johnstone [5] or Awodey [1]. □

It is also possible to construct a λ-theory from a ccc.

Definition 6.12. If $\mathcal{C}$ is a small cartesian closed category then its *internal language* $\mathbb{L}(\mathcal{C})$ is the smallest λ-theory satisfying the following conditions:

(1) For every object A of $\mathcal{C}$ there is a basic type $\ulcorner A \urcorner$ in $\mathbb{L}(\mathcal{C})$, and this assignment is injective;
(2) For every morphism $f : A \to B$ there is a basic constant
$$* : 1 \vdash \ulcorner f \urcorner : \ulcorner A \urcorner \to \ulcorner B \urcorner,$$
and this assignment is injective;
(3) There is a collection of additional constants whose purpose is to ensure that the product and function types in $\mathbb{L}(\mathcal{C})$ coincide with the products and exponentials in $\mathcal{C}$; I will denote them all by ι since they represent identity morphisms, and I omit the subscripts that ought to decorate the ι:
   (a) $* : 1 \vdash \iota : 1 \to \ulcorner 1 \urcorner$;
   (b) $* : 1 \vdash \iota : \ulcorner A \urcorner \times \ulcorner B \urcorner \to \ulcorner A \times B \urcorner$, for each pair of objects A and B;
   (c) $* : 1 \vdash \iota : (\ulcorner A \urcorner \to \ulcorner B \urcorner) \to \ulcorner B^A \urcorner$, for each pair of objects A and B.
(4) The theory $\mathbb{L}(\mathcal{C})$ has the following axioms to make these constants work as desired:
   (a) For each object A there is the axiom $x : \ulcorner A \urcorner \vdash \ulcorner id_A \urcorner x = x : \ulcorner A \urcorner$, and for each commutative triangle



there is the axiom $x : \ulcorner A \urcorner \vdash \ulcorner g \urcorner(\ulcorner f \urcorner x) = \ulcorner h \urcorner x : \ulcorner C \urcorner$;

(b) There is the axiom $t:\ulcorner 1\urcorner \vdash \iota * = t:\ulcorner 1\urcorner$, and for each pair of objects A and B there are the axioms

$$z:\ulcorner A \times B\urcorner \vdash \iota\langle \ulcorner \pi_1\urcorner z, \ulcorner \pi_2\urcorner z\rangle = z:\ulcorner A \times B\urcorner,$$
$$w:\ulcorner A\urcorner \times \ulcorner B\urcorner \vdash \langle \ulcorner \pi_1\urcorner(\iota w), \ulcorner \pi_2\urcorner(\iota w)\rangle = w:\ulcorner A\urcorner \times \ulcorner B\urcorner;$$

(c) For each pair of objects A and B there are the axioms

$$f:\ulcorner B^A\urcorner \vdash \iota(\lambda x:\ulcorner A\urcorner.\ulcorner ev\urcorner(\iota\langle f, x\rangle)) = f:\ulcorner B^A\urcorner,$$
$$f:\ulcorner A\urcorner \to \ulcorner B\urcorner \vdash \lambda x:\ulcorner A\urcorner.\ulcorner ev\urcorner(\iota\langle \iota f, x\rangle) = f:\ulcorner A\urcorner \to \ulcorner B\urcorner.$$

In Section 7 we will see some of the desirable properties of these internal languages. Johnstone [5, Section D4.2], never one to fly in the face of space constraints, goes into greater detail. (Awodey [1, Section 2.4] gives a more accessible treatment.)

In this section I have talked mainly about the relationship between ccc's and λ-theories; see Lambek [6] for a construction of a ccc from a PIP theory. Lambek follows a method similar to the one suggested by Remark 6.9, giving a certain equivalence relation over proofs.

7. Curry-Howard-Lambek, categorically

We are now in a position to discuss the third part of the diagram presented in the introduction. The process is essentially a categorification of Section 6, so I will proceed with brevity.

Definition 7.1. Ccc is the category whose objects are small cartesian closed categories and whose morphisms are functors which preserve finite products and exponentials. These are known as *cartesian closed functors*.

We would like a pair of functors between λ**Thr** and **Ccc**. We already have the construction of a syntactic category from a λ-theory, and this extends to a functor $\mathcal{S} : \lambda\mathbf{Thr} \to \mathbf{Ccc}$ because a translation $\tau : \mathbb{L} \to \mathbb{M}$ induces a ccc functor in the obvious way such that the diagram

$$\begin{array}{ccc} \mathbb{L} & \xrightarrow{\mathcal{S}} & \mathcal{S}(\mathbb{L}) \\ {\scriptstyle\tau}\downarrow & & \downarrow{\scriptstyle\mathcal{S}(\tau)} \\ \mathbb{M} & \xrightarrow{\mathcal{S}} & \mathcal{S}(\mathbb{M}) \end{array}$$

commutes.

Similarly, the construction of an internal language from a ccc extends to a functor $\mathbb{L} : \mathbf{Ccc} \to \lambda\mathbf{Thr}$ because a ccc functor $F : \mathcal{C} \to \mathcal{D}$ induces a translation making the diagram

$$\begin{array}{ccc} \mathcal{C} & \xrightarrow{\mathbb{L}} & \mathbb{L}(\mathcal{C}) \\ {\scriptstyle F}\downarrow & & \downarrow{\scriptstyle\mathbb{L}(F)} \\ \mathcal{D} & \xrightarrow{\mathbb{L}} & \mathbb{L}(\mathcal{C}) \end{array}$$

commute.

Theorem 7.2. Ccc *is equivalent to* λ**Thr** *"up to equivalence." This means that a cartesian closed category* $\mathcal{C}$ *is equivalent to* $\mathcal{S}(\mathbb{L}(\mathcal{C}))$*, and a* λ*-theory* $\mathbb{M}$ *is 'Morita-equivalent' to* $\mathbb{L}(\mathcal{S}(\mathbb{M}))$ *in the sense that there is a natural bijection between isomorphism classes of models of* $\mathbb{M}$ *and* $\mathbb{L}(\mathcal{S}(\mathbb{M}))$ *in arbitrary ccc's.*

Proof. See Awodey [1, Theorem 2.4.4]. The equivalence is only up to equivalence, rather than up to isomorphism, because $\mathcal{S}(\mathbb{L}(\mathcal{C}))$ will have many more objects than $\mathcal{C}$; however each of these objects will be isomorphic to an object of the form $\ulcorner A \urcorner$ where A is an object of $\mathcal{C}$. □

We have now established that **PIPThr** is equivalent to a full subcategory of λ**Thr**, which in turn is equivalent (up to equivalence) to **Ccc**:

$$\begin{array}{ccc} \mathbf{PIPThr} & \overset{\sim}{\hookrightarrow} & \lambda\mathbf{Thr} \\ & & \mathbb{L} \uparrow \downarrow \mathcal{S} \\ & & \mathbf{Ccc} \end{array}$$

Further work is required to investigate the missing third side of the triangle, but I conjecture that it involves a similar sort of 'Morita equivalence' of PIP theories.

Acknowledgments. It is a pleasure to thank my mentors, Matthew Wright and Jonathan Stephenson, for their patient advice and helpful suggestions. I would also like to thank my family for tolerating my work sprawled over the dining room table.

References

[1] S. Awodey and A. Bauer, *Lecture notes: introduction to categorical logic*, Draft, 2003. http://www.andrew.cmu.edu/user/awodey/catlog/notes/notes2.pdf
[2] H. Barendregt, *The lambda calculus: its syntax and semantics*, North-Holland, 1984.
[3] R. Goldblatt, *Topoi: the categorial analysis of logic*, North-Holland, 1984.
[4] W. A. Howard, *The formulae-as-types notion of construction*, in To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism (P. Seldin and R. Hindley, eds.), Academic Press, 1980, pp. 479-490.
[5] P. T. Johnstone, *Sketches of an elephant: a topos theory compendium, vol. II*, Clarendon Press, 2002.
[6] J. Lambek, *Deductive systems and categories, III*, in Toposes, Algebraic Geometry, and Logic (F. W. Lawvere, ed.), Springer-Verlag, 1972, pp. 57-82.
[7] J. Lambek, *From λ-calculus to cartesian closed categories*, in To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism (P. Seldin and R. Hindley, eds.), Academic Press, 1980, pp. 375-402.
[8] D. Scott, *Continuous lattices*, in Toposes, Algebraic Geometry, and Logic (F. W. Lawvere, ed.), Springer-Verlag, 1972, pp. 97-136.
[9] D. Scott, *Relating theories of the λ-calculus*, in To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism (P. Seldin and R. Hindley, eds.), Academic Press, 1980, pp. 403-450.