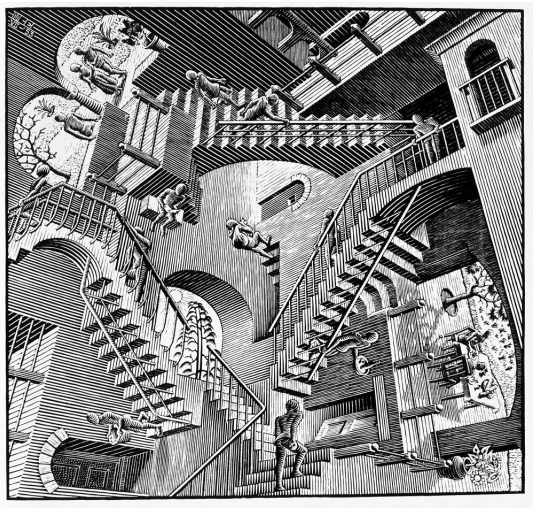
CTCC Seminar Notes:
The Curry-Howard-Lambek Correspondence

Lorenzo Pace

May 02, 2024



Abstract

The Curry-Howard-Lambek correspondence is a beautiful link between logic, computation, and category theory. We will review the necessary tools from the three disciplines and illustrate the correspondence. Finally, we shall define the categorical semantics of the $\lambda^{\to\times}$-calculus and prove some key results regarding its properties.

Contents

Introduction

"Algorithms are the computational content of proofs."

–Robert Harper

The scope of the Correspondence The Curry-Howard correspondence serves as a *Rosetta stone* that allows one to interpret computations as proofs and vice versa. In its original formulation, it provided a bridge between type theory and *constructive* logic. More recent developments [3] show that there exists a similar correspondence between a variant of the λ-calculus (extended with *control operators*) and *classical* logic. We shall not delve into this, and will only work with the original formulation.

A Brief History of the Correspondence The origins of the result lie in some key observations made by Haskell Curry and William Howard in the span of around thirty years. The culmination of these observations was Howard's realization that intuitionistic natural deduction could be intepreted as a typed variant of the λ-calculus. This sparked a new area of research, which lead to new formal systems which could be interpreted both as proof systems and as functional programming languages. Among these, there is the Calculus of Constructions, which is the basis of the Coq proof assistant.

Cartesian Closed Categories Johachim Lambek [4], in the 1970's, showed that proofs and computations shared a common equational theory, that of Cartesian Closed Categories. We will provide a categorical semantics of the simply typed λ-calculus, show its soundness and discuss its completeness.

1 Logic

1.1 Constructive Logic

In classical logic, there are two ways to show that some property holds. We can either build a constructive proof, that is, start from a collection of axioms and apply a sequence of inference rules until we reach the statement of the desired property, or we show that the negation of the property leads to contradiction.

In the logic we consider, called *constructive* or *intuitionistic* logic, we only want the constructive proofs, and we do not assume the rules that allow to prove by contradiction, which are:

- The *law of excluded middle*:
$$\vdash A \vee \neg A$$
- Double negation elimination:
$$\neg\neg A \vdash A$$

1.2 The Natural Deduction System

We consider the fragment of propositional logic conjunction $\wedge$ and implication $\supset$. This is called the *natural deduction system* for $\wedge, \supset$. Its axiom and inference rules follow:

Identity	Conjunction	Implication
$\dfrac{}{\Gamma, A \vdash A}$ Id	$\dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$ $\wedge$ intro	$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B}$ $\supset$ intro
	$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}$ $\wedge$ elim$_1$	$\dfrac{\Gamma \vdash A \supset B \qquad \Gamma \vdash A}{\Gamma \vdash B}$ $\supset$ elim
	$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$ $\wedge$ elim$_2$	

Definition 1.1 (Admissibility)**.** We say that a rule is *admissible* if whenever there exists a proof of its premises, there also exists a proof of its conclusions.

2 Computation

The reader is assumed to be familiar with the λ-calculus [2], and we will recall some concepts regarding the *simply typed λ-calculus with products*.

2.1 Simply Typed λ-calculus with products

The extension of the λ-calculus we consider allows us to assign types to the λ-terms. Let us assume a set of base types, ranged over by b. These types can be combined in such a way to obtain *function types* and *product types*, which will be assigned to functions and *pairs* of terms respectively. The syntax is:

$$
\begin{aligned}
T, U &::= b \mid T \to U \mid T \times U && \text{(Types)} \\
t, u &::= x \mid tu \mid \lambda x.t \mid \langle t, u \rangle \mid \pi_1 u \mid \pi_2 u && \text{(Terms)} \\
\Gamma &::= \emptyset \mid x : T, \Gamma \qquad ((x : \_) \notin \Gamma) && \text{(Typing contexts)}
\end{aligned}
$$

Type Judgement The assertion "term t has type T in context Γ" is written as:

$$\Gamma \vdash t : T$$

And a *typed term* is a triple (Γ, t, T), s.t. the judgement above can be derived from the following rules:

Variable	Product	Function
$\dfrac{}{\Gamma, x : T \vdash x : T}$	$\dfrac{\Gamma \vdash t : T \qquad \Gamma \vdash u : U}{\Gamma \vdash \langle t, u \rangle : T \times U}$	$\dfrac{\Gamma, x : U \vdash t : T}{\Gamma \vdash \lambda x.t : U \to T}$
	$\dfrac{\Gamma \vdash v : T \times U}{\Gamma \vdash \pi_1 v : T}$	$\dfrac{\Gamma \vdash t : U \to T \qquad \Gamma \vdash u : U}{\Gamma \vdash t\, u : T}$
	$\dfrac{\Gamma \vdash v : T \times U}{\Gamma \vdash \pi_2 v : U}$	

Notice anything? Indeed, these rules are identical to the ones we defined for the Natural Deduction System, by equating conjunction with $\times$ and implication with $\to$. This correspondence is what's called the *Curry-Howard Correspondence*.

$$
\begin{aligned}
\text{Natural Deduction System} &\longleftrightarrow \lambda^{\to \times}\text{-calculus} \\
\text{Formulas} &\longleftrightarrow \text{Types} \\
\text{Proofs} &\longleftrightarrow \text{Terms} \\
\text{Proof transformations} &\longleftrightarrow \text{Term Reductions}
\end{aligned}
$$

Before moving on to categories, we introduce the semantics of the lambda calculus, of which we will later give a categorical formulation.

Semantics The semantics of simply-typed lambda calculus is given by the following relations:

- β-reduction:

  $$
  \begin{aligned}
  (\lambda x.t)u &\to_\beta t[u/x] \\
  \pi_1 \langle t, u \rangle &\to_\beta t \\
  \pi_2 \langle t, u \rangle &\to_\beta u
  \end{aligned}
  $$

  whose symmetric, transitive, reflexive closure is called β-conversion.

- η-conversion:

  $$
  \begin{aligned}
  t &=_\eta \lambda x.tx && \text{for function types with } x \notin FV(t) \\
  v &=_\eta \langle \pi_1 v, \pi_2 v \rangle && \text{for product types}
  \end{aligned}
  $$

- λ-conversion ($=_\lambda$), defined as the transitive closure of $(=_\beta) \cup (=_\eta)$

Strong Normalization It can be shown that these rules form a *strongly normalizing* rewriting system, that is, for every term t it is possible to reach a normal form (an irreducible expression) after applying the β-reduction rule a finite number of times. This implies that every reduction eventually terminates and results in a normal form. Note that this was not the case in the untyped λ calculus, as there were some combinators such as Ω that led to infinite derivations. This is because Ω and other similar combinators use self-application. This is not a problem with types because self-application is not typeable with the provided rules. In light of the Curry-Howard Correspondence, term in normal form corresponds to a proof in which all lemmas have been eliminated.

Proof as computation

- A proof of an implication $A \supset B$ can be seen as a program that transforms any proof of A into a proof of B.
- A proof of the conjunction $A \vee B$ can be seen as a pair of proofs, one for A and one for B.

3 Categories

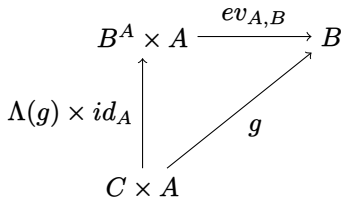
3.1 Exponentials and Cartesian Closed Categories

Definition 3.1. A *cartesian closed category* is one with a terminal object, products and exponentials. The reader is assumed to know what a terminal object and products are, but exponentials will be quickly reviewed.

Definition 3.2. A category $\mathcal{C}$ is said to *have exponentials* if for all objects A and B there exists an object B^A of $\mathcal{C}$ and a morphism:

$$ev_{A,B} : B^A \times A \to B$$

Such that for all morphisms $g : C \times A \to B$ there exists a unique morphism $\Lambda(g) : C \to B^A$ such that the following diagram commutes:



Computational Interpretation For now, let $\mathcal{C}$ be Set. As the notation suggest, the set B^A is the set of functions from A to B. We can thus interpret the morphism:

$$ev_{A,B} : B^A \times A \to B$$

as an *evaluation function*, that takes a function f and an element a of its domain, and returns the result of applying f to a, that is:

$$ev_{A,B} : (f, a) \mapsto f(a)$$

Writing in functional language syntax to avoid stacking too many superscipts, the type of Λ is:

$$\Lambda : ((C \times A) \to B) \to (C \to A \to B)$$

Which is (up to renaming the type variables) precisely the signature of the `curry` function in Haskell:

```haskell
curry :: ((a, b) -> c) -> a -> b -> c
```

As the reader likely knows, *currying* (i.e. applying the curry function) is used to express functions of more than one parameter in the λ-calculus. As an example:

Uncurried Version	Curried Version
$\text{sum} = \lambda(x, y).(x + y)$	$\text{sum}_C = \lambda x.\lambda y.(x + y)$

Indeed, if we start from a pair of natural numbers $(a, b) \in \mathbb{N}^2$ and we apply the function sum_C to the first argument, obtaining the pair:

$$(\lambda y.(a + y), b)$$

and then apply the evaluation function to this pair, we obtain the result of the sum $a + b$, which is precisely what we would have obtained by just applying the function sum to the initial pair (a, b).

It is obvious that this holds for any function of type $(C \times A) \to B$, and it is precisely what the commutative diagram above says. Therefore, Λ is exactly the `curry` function.

3.2 The Correspondence

We will now illustrate the correspondence between logic and CCCs. Let $\mathcal{C}$ be a cartesian closed category. We shall interpret formulas (or types) as objects in $\mathcal{C}$, and proofs of B given A as morphisms. Since there are all products, we can transform:

$$A_1, \ldots, A_k \vdash A$$

into:

$$f : A_1 \times \ldots \times A_k \to A$$

Then, the correspondence is given as follows:

Axiom	$\dfrac{}{\Gamma, A \vdash A}\ \mathsf{Id}$	$\overline{\pi_2 : \Gamma \times A \longrightarrow A}$
Conjunction	$\dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge\mathsf{I}$	$\dfrac{f : \Gamma \longrightarrow A \qquad g : \Gamma \longrightarrow B}{\langle f, g \rangle : \Gamma \longrightarrow A \times B}$
	$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge\mathsf{E}_1$	$\dfrac{f : \Gamma \longrightarrow A \times B}{\pi_1 \circ f : \Gamma \longrightarrow A}$
	$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge\mathsf{E}_2$	$\dfrac{f : \Gamma \longrightarrow A \times B}{\pi_2 \circ f : \Gamma \longrightarrow B}$
Implication	$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B}\ \supset\mathsf{I}$	$\dfrac{f : \Gamma \times A \longrightarrow B}{\Lambda(f) : \Gamma \longrightarrow (A \Rightarrow B)}$
	$\dfrac{\Gamma \vdash A \supset B \quad \Gamma \vdash A}{\Gamma \vdash B}\ \supset\mathsf{E}$	$\dfrac{f : \Gamma \longrightarrow (A \Rightarrow B) \quad g : \Gamma \to A}{\mathsf{ev}_{A,B} \circ \langle f, g \rangle : \Gamma \longrightarrow B}$

Note that it is ok to map a *set* of premises to a *tuple* of premises, because the each n-ary product is isomorphic to each of its permutations (CCCs are *symmetric*), hence ordering does not matter.

3.3 Categorical Semantics of $\lambda^{\to\times}$-calculus

In this part, we define a function $[\![-]\!]$ that maps every typed term:

$$\Gamma \vdash t : T$$

to a $\mathcal{C}$-morphism $[\![t]\!] : [\![\Gamma]\!] \to [\![T]\!]$.

Definition 3.3 (Categorical semantics of $\lambda^{\to\times}$-calculus)**.** We define the semantics recursively, assuming that each base type b is mapped to object $\tilde{b}$.

- Definition on types:

$$[\![b]\!] := \tilde{b}, \qquad [\![T \times U]\!] := [\![T]\!] \times [\![U]\!], \qquad [\![T \to U]\!] := [\![U]\!]^{[\![T]\!]}$$

- Definition on typed terms:

$$\frac{}{[\![\Gamma, x : T \vdash x : T]\!] := \pi_2 : [\![\Gamma]\!] \times [\![T]\!] \longrightarrow [\![T]\!]}$$

$$\frac{[\![\Gamma \vdash t : T \times U]\!] = f : [\![\Gamma]\!] \longrightarrow [\![T]\!] \times [\![U]\!]}{[\![\Gamma \vdash \pi_1 t : T]\!] := [\![\Gamma]\!] \xrightarrow{f} [\![T]\!] \times [\![U]\!] \xrightarrow{\pi_1} [\![T]\!]}$$

$$\frac{[\![\Gamma \vdash t : T]\!] = f : [\![\Gamma]\!] \longrightarrow [\![T]\!] \qquad [\![\Gamma \vdash u : U]\!] = g : [\![\Gamma]\!] \longrightarrow [\![U]\!]}{[\![\Gamma \vdash \langle t, u \rangle : T \times U]\!] := [\![\Gamma]\!] \xrightarrow{\langle f,g \rangle} [\![T]\!] \times [\![U]\!]}$$

$$\frac{[\![\Gamma, x : T \vdash t : U]\!] = f : [\![\Gamma]\!] \times [\![T]\!] \longrightarrow [\![U]\!]}{[\![\Gamma \vdash \lambda x.t : T \to U]\!] := \Lambda(f) : [\![\Gamma]\!] \longrightarrow ([\![T]\!] \Rightarrow [\![U]\!])}$$

$$\frac{[\![\Gamma \vdash t : T \to U]\!] = f \qquad [\![\Gamma \vdash u : T]\!] = g}{[\![\Gamma \vdash t\, u : U]\!] := [\![\Gamma]\!] \xrightarrow{\langle f,g \rangle} ([\![T]\!] \Rightarrow [\![U]\!]) \times [\![T]\!] \xrightarrow{\mathsf{ev}} [\![U]\!]}$$

We shall now discuss two key properties of this semantics. In order to do so, we first need to discuss a few useful results.

3.4 Preliminary notions

3.4.1 Properties of Λ

Proposition 3.4. Given any function $f : A \to C^B$, the following holds:

$$f = \Lambda((f \times id_B); ev_{B,C})$$

Proof. By definition of Λ, the following diagram commutes:

$$A \times B \;\underset{f \times id_B}{\overset{\Lambda((f \times id_B); ev_{B,C}) \times id_B}{\rightrightarrows}}\; C^B \times B \xrightarrow{ev_{B,C}} C$$

This diagram is obtained by simply plugging $(f \times id_B); ev_{B,C}$ in place of g in the definition of Λ. □

Naturality of Λ The transformation Λ can be shown to be natural in all three variables. Moreover, it is a *natural isomorphism.* See the appendix for more details on this.

3.4.2 Substitution Lemma

Simultaneous substitution In the following, interpret $t[t_1/x_1, \ldots, t_k/x_k]$ as the *simultaneous substitution* of $x_1, \ldots, x_k \in FV(t)$ with terms $t_1, \ldots, t_k$. The precise definition is given by induction on the terms, and can be found on in the appendix. For the sake of intuition, you can think of simultaneous substitution as the following:

$$t[t_1/x_1, \ldots, t_k/x_k] = t[t_1/x_1] \ldots [t_k/x_k]$$

Now we can state the substitution lemma.

Lemma 3.5 (Substitution Lemma)**.** Let $\Gamma = x_1 : T_1, \ldots, x_k : T_k$. Then, given terms:

$$\Gamma \vdash t : T \qquad \Gamma \vdash t_i : T_i \quad i \in \{1, \ldots, k\}$$

Then, the following holds:

$$[\![t[t_1/x_1, \ldots, t_k/x_k]]\!] = \langle [\![t_1]\!], \ldots, [\![t_k]\!] \rangle; [\![t]\!]$$

The proof of this lemma, by structural induction on t, can be found in [1]. A proof sketch is also presented in the appendix.

3.5 Soundness

First, we will show that the semantics is sound, that is, it preserves λ-equivalences:

$$t =_\lambda u \implies [\![t]\!] = [\![u]\!]$$

Notation In some of the following proofs, we shall use the notation:

$$[\![\Gamma]\!] \xrightarrow{[\![a]\!]} [\![A]\!]$$

In place of $[\![\Gamma \vdash a : A]\!] : [\![\Gamma]\!] \to [\![A]\!]$. This will be used to focus on how the changes in domain of the composed morphisms mimick the types of the corresponding terms.

Proposition 3.6. The semantics preserves β-conversion.

Proof.

- The first rule is: $(\lambda x.t)u \to_\beta t[u/x]$. We thus prove:

$$\begin{aligned}
[\![\Gamma \vdash (\lambda x.t)u : B]\!] &= [\![\Gamma]\!] \xrightarrow{\langle [\![\lambda x.t]\!], [\![u]\!] \rangle} [\![B]\!]^{[\![A]\!]} \times [\![A]\!] \xrightarrow{ev} [\![B]\!] && \text{(Def. } [\![-]\!]) \\
&= [\![\Gamma]\!] \xrightarrow{\langle \Lambda [\![t]\!], [\![u]\!] \rangle} [\![B]\!]^{[\![A]\!]} \times [\![A]\!] \xrightarrow{ev} [\![B]\!] && \text{(Def. } [\![-]\!]) \\
&= [\![\Gamma]\!] \xrightarrow{\langle id_{[\![\Gamma]\!]}, [\![u]\!] \rangle} [\![\Gamma]\!] \times [\![A]\!] \xrightarrow{\Lambda [\![t]\!] \times id_{[\![A]\!]}} [\![B]\!]^{[\![A]\!]} \times [\![A]\!] \xrightarrow{ev} [\![B]\!] && \text{(Prop. A.3)} \\
&= [\![\Gamma]\!] \xrightarrow{\langle id_{[\![\Gamma]\!]}, [\![u]\!] \rangle} [\![\Gamma]\!] \times [\![A]\!] \xrightarrow{[\![t]\!]} [\![B]\!] && \text{(Def. } \Lambda) \\
&= [\![\Gamma]\!] \xrightarrow{[\![t[u/x]]\!]} [\![B]\!] && \text{(Subst. Lemma)}
\end{aligned}$$

- The second rule is: $\pi_1 \langle t, u \rangle \to_\beta t$

$$\begin{aligned}
[\![\Gamma \vdash \pi_1 \langle t, u \rangle : A]\!] &= [\![\langle t, u \rangle]\!]; \pi_1 && \text{(Def. } [\![-]\!]) \\
&= \langle [\![t]\!], [\![u]\!] \rangle; \pi_1 && \text{(Def. } [\![-]\!]) \\
&= [\![t]\!] && \text{(Prop. } \times)
\end{aligned}$$

- The proof for the third rule ($\pi_2 \langle t, u \rangle \to_\beta u$) is analogous to the previous point.

□

Proposition 3.7. The semantics preserves η-conversion.

Proof.

- First rule: $t =_\eta \lambda x.tx$

$$\begin{aligned}
[\![\Gamma \vdash \lambda x.tx : A \to B]\!] &= [\![\Gamma]\!] \xrightarrow{\Lambda [\![tx]\!]} [\![B]\!]^{[\![A]\!]} && \text{(Def } [\![-]\!]) \\
&= \Lambda([\![\Gamma]\!] \times [\![A]\!] \xrightarrow{[\![tx]\!]} [\![B]\!]) && \text{(Rewriting)} \\
&= \Lambda([\![\Gamma]\!] \times [\![A]\!] \xrightarrow{\langle [\![t]\!], [\![x]\!] \rangle} [\![B]\!]^{[\![A]\!]} \times [\![A]\!] \xrightarrow{ev_{A,B}} [\![B]\!]) && \text{(Def } [\![-]\!]) \\
&= \Lambda([\![\Gamma]\!] \times [\![A]\!] \xrightarrow{[\![t]\!]' \times id_A} [\![B]\!]^{[\![A]\!]} \times [\![A]\!] \xrightarrow{ev_{A,B}} [\![B]\!]) && \text{(Prop. A.2)} \\
&= [\![\Gamma]\!] \xrightarrow{[\![t]\!]'} [\![B]\!]^{[\![A]\!]} && \text{(Prop. 3.4)}
\end{aligned}$$

  Where $[\![t]\!]'$ is the morphism corresponding to: $\Gamma \vdash t : B$ and $[\![t]\!]$ corresponds to $\Gamma, (x : A) \vdash t : B$.

- Second rule: $v =_\eta \langle \pi_1 v, \pi_2 v \rangle$

$$\begin{aligned}
[\![\Gamma \vdash \langle \pi_1 v, \pi_2 v \rangle : A \times B]\!] &= \langle [\![t]\!]; \pi_1, [\![t]\!]; \pi_2 \rangle && \text{(Def. } [\![-]\!]) \\
&= [\![t]\!] && \text{(Prop. } \times)
\end{aligned}$$

□

3.6 Completeness?

We briefly touch on the matter of completeness. As it turns out, for a generic CCC, there may exist equalities that are not reflected by the semantic translation, i.e.

$$[\![t]\!] = [\![u]\!] \quad \text{yet} \quad t \neq_\lambda u$$

In spite of this, a CCC C_λ can be constructed in such a way that equalities between arrows correspond exactly to λ-conversions.

Definition 3.8. We define a family of equivalence relations:

$$(x,t) \sim_{T,U} (y,u) \iff x : T \vdash t : U, y : T \vdash u : U \text{ are derivable and } t =_\lambda u[x/y]$$

$$(\bullet,t) \sim_{\bullet,U} (\bullet,u) \iff \vdash t : U \text{ is derivable and } t =_\lambda u$$

We denote as $[(x,t)]_{T,U}$ as the equivalence class of (x,t) under $\sim_{T,U}$, and $[(\bullet,t)]_{\bullet,U}$ as the equivalence class of $(\bullet,T)$ under $\sim_{\bullet,U}$.

The construction is as follows:

- The objects of the category C_λ are:

  $$Ob(C_\lambda) = \{\mathbf{1}\} \cup \{\widetilde{T} \mid T \text{ is a } \lambda\text{-type}\}$$

  That is, the set of λ-types augmented with a terminal object.

- The homsets contain equivalence classes and terminal arrows $\tau_A : A \to \mathbf{1}$

  $$\begin{aligned}
  \mathrm{Hom}_{C_\lambda}(\widetilde{T},\widetilde{U}) &:= \{[(x,t)]_{T,U} \mid x : T \vdash t : U \text{ is derivable}\} \\
  \mathrm{Hom}_{C_\lambda}(\mathbf{1},\widetilde{U}) &:= \{[(\bullet,t)]_{\bullet,U} \mid \vdash t : U \text{ is derivable}\} \\
  \mathrm{Hom}_{C_\lambda}(A,\mathbf{1}) &:= \{\tau_A\}
  \end{aligned}$$

- The identities are $id_{\widetilde{T}} = [(x,x)]_{T,T}$ and $id_{\mathbf{1}} = \tau_{\mathbf{1}}$.

- Arrow composition is defined by:

  $$\begin{aligned}
  [(y,u)]_{T,U}; [(x,t)]_{U,V} &:= [(y, t[u/x])]_{T,V} \\
  [(\bullet,t)]_{\bullet,U}; [(x,t)]_{U,V} &:= [(\bullet, t[u/x])]_{\bullet,V} \\
  \tau_A; [(\bullet,t)]_{\bullet,T} &:= \begin{cases} [(y,t)]_{U,T} & \text{if } A = \widetilde{U} \\ [(\bullet,t)]_{\bullet,T} & \text{if } A = \mathbf{1} \end{cases} \\
  h; \tau_B &:= \tau_A \qquad\qquad (h \in \mathrm{Hom}_{C_\lambda}(A,B))
  \end{aligned}$$

It can be shown that this is in fact a category, and that it has finite products and exponentials. These proofs can be found in [1]. Since this is a CCC, it is a sound model of the simply typed λ-calculus described earlier. Additionally, we have:

$$[\![\Gamma \vdash t : T]\!] = [(x, t[\pi_i(x)/x_i])]_{\Gamma,T} \qquad i \in \{1 \ldots n\}$$

Where:

$$\Gamma = \{x_1 : T_1, \ldots, x_n : T_n\}, \qquad x \notin \Gamma, \qquad x = T_1 \times \ldots \times T_n.$$

Since the denotation of a term corresponds to an equivalence class over λ-conversion, every λ-equivalent term shall be mapped to the same morphism. We therefore also have completeness.

A Additional material

A.1 Naturality of Λ

It turns out that:

$$\Lambda : \mathrm{Hom}(A \times B, C) \Rightarrow \mathrm{Hom}(A, C^B)$$

defines an isomorphism that is natural in its parameters. We do not show that it defines an isomorphism, but we show the naturality in A, which is simply done by showing that the following diagram commutes:

$$\begin{array}{ccc} \mathrm{Hom}(A \times B, C) & \xrightarrow{\Lambda_A} & \mathrm{Hom}(A, C^B) \\ \Big\downarrow{\scriptstyle (g \times id_B); -} & & \Big\downarrow{\scriptstyle g; -} \\ \mathrm{Hom}(A' \times B, C) & \xrightarrow[\Lambda'_A]{} & \mathrm{Hom}(A', C^B) \end{array}$$

Proof. Let $f : A \times B \to C$ and $g : A \to A'$. We show that $g; \Lambda f = \Lambda((g \times id_B); f)$.

$$\begin{aligned} g; \Lambda f &= \Lambda(((g; \Lambda f) \times id_B); ev_{B;C}) && \text{(Prop. 3.4)} \\ &= \Lambda((g \times id_B); (\Lambda f \times id_B); ev_{B,C}) && \text{(Prop. } \times\text{)} \\ &= \Lambda((g \times id_B); f) && \text{(Def. } \Lambda\text{)} \end{aligned}$$

□

A.2 The substitution lemma

The following is the formal definition of simultaneous substitution.

Definition A.1. Let $\Gamma = x_1 : T_1, \ldots, x_k : T_k$. Given typed terms:

$$\Gamma \vdash t : T \qquad \Gamma \vdash t_i : T_i \quad 1 \leq i \leq k$$

such that the x_i are free variables in t, we define $t[\mathbf{t}/\mathbf{x}] \equiv t[t_1/x_1, \ldots, t_k/x_k]$ recursively as:

$$\begin{aligned} x_i[\boldsymbol{t}/\boldsymbol{x}] &:= t_i \\ (\pi_i\, t)[\boldsymbol{t}/\boldsymbol{x}] &:= \pi_i(t[\boldsymbol{t}/\boldsymbol{x}]) \\ \langle t, u \rangle[\boldsymbol{t}/\boldsymbol{x}] &:= \langle t[\boldsymbol{t}/\boldsymbol{x}], u[\boldsymbol{t}/\boldsymbol{x}] \rangle \\ (t\, u)[\boldsymbol{t}/\boldsymbol{x}] &:= (t[\boldsymbol{t}/\boldsymbol{x}])(u[\boldsymbol{t}/\boldsymbol{x}]) \\ (\lambda x.\, t)[\boldsymbol{t}/\boldsymbol{x}] &:= \lambda x.\, t[\boldsymbol{t}, x/\boldsymbol{x}, x]\,. \end{aligned}$$

Where the notation in the last line defines that the variable x, which is free in the subterm t, should be preserved, i.e. substituted with itself.

Proof sketch of The Substitution Lemma We now provide a sketch of the proof of Lemma 3.4.2:

$$[\![t[t_1/x_1, \ldots, t_k/x_k]]\!] = \langle [\![t_1]\!], \ldots, [\![t_k]\!] \rangle; [\![t]\!]$$

Proof sketch. We prove the base case and the interesting inductive cases.

- Let $t = x_i$:

$$\begin{aligned} [\![x_i[\mathbf{t}/\mathbf{x}]]\!] &= [\![t]\!]_i \\ &= \langle [\![t_1]\!], \ldots, [\![t_k]\!] \rangle; \pi_i \\ &= \langle [\![t_1]\!], \ldots, [\![t_k]\!] \rangle; [\![x_i]\!] && \text{(Def. } [\![-]\!]\text{)} \end{aligned}$$

- Let $t = uv$:

$$\begin{aligned}
[\![uv[\mathbf{t}/\mathbf{x}]]\!] &= [\![(u[\mathbf{t}/\mathbf{x}])(v[\mathbf{t}/\mathbf{x}])]\!] && \text{(Def. subst.)}\\
&= \langle [\![u[\mathbf{t}/\mathbf{x}]]\!], [\![v[\mathbf{t}/\mathbf{x}]]\!] \rangle ; ev && \text{(Def. } [\![-]\!]\text{)}\\
&= \langle \langle [\![t_1]\!], \ldots, [\![t_k]\!] \rangle ; [\![u]\!], \langle [\![t_1]\!], \ldots, [\![t_k]\!] \rangle ; [\![v]\!] \rangle ; ev && \text{(Ind. Hp.)}\\
&= \langle [\![t_1]\!], \ldots, [\![t_k]\!] \rangle ; \langle [\![u]\!], [\![v]\!] \rangle ; ev && \text{(Prop. } \times\text{)}\\
&= \langle [\![t_1]\!], \ldots, [\![t_k]\!] \rangle ; [\![uv]\!] && \text{(Def. } [\![-]\!]\text{)}
\end{aligned}$$

- Let $t = \lambda x.u$:

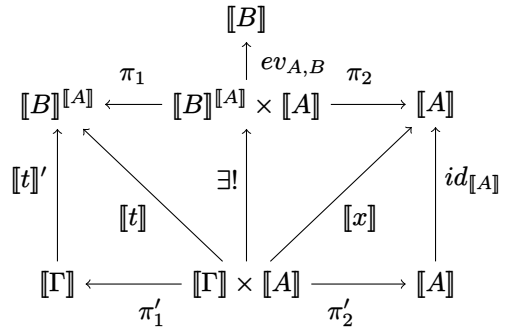
$$\begin{aligned}
[\![\Gamma \vdash (\lambda x.u)[\mathbf{t}/\mathbf{x}] : A \to B]\!] &= [\![\Gamma \vdash (\lambda x.u[\mathbf{t}, x/\mathbf{x}, x]) : A \to B]\!] && \text{(Def subst.)}\\
&= \Lambda([\![\Gamma]\!] \times [\![A]\!] \xrightarrow{[\![u[\mathbf{t},x/\mathbf{x},x]]\!]} [\![B]\!]) && \text{(Def. } [\![-]\!]\text{)}\\
&= \Lambda([\![\Gamma]\!] \times [\![A]\!] \xrightarrow{\langle [\![t_1]\!], \ldots, [\![t_k]\!], id_A \rangle} [\![\Gamma]\!] \times [\![A]\!] \xrightarrow{[\![u]\!]} [\![B]\!]) && \text{(Ind. Hp.)}\\
&= \Lambda([\![\Gamma]\!] \times [\![A]\!] \xrightarrow{\langle [\![t_1]\!], \ldots, [\![t_k]\!] \rangle \times id_A} [\![\Gamma]\!] \times [\![A]\!] \xrightarrow{[\![u]\!]} [\![B]\!]) && \text{(Prop. } \times\text{)}\\
&= \langle [\![t_1]\!], \ldots, [\![t_k]\!] \rangle ; \Lambda([\![u]\!]) && \text{(Naturality } \Lambda\text{)}\\
&= \langle [\![t_1]\!], \ldots, [\![t_k]\!] \rangle ; [\![\lambda x.u]\!] && \text{(Def. } [\![-]\!]\text{)}
\end{aligned}$$

- Projections and pairs are not as interesting, and are left to the reader.

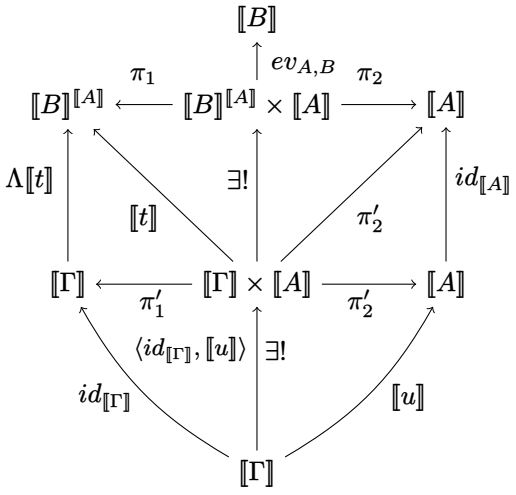
□

A.3 Reference diagrams

Proposition A.2. The following diagram commutes:



Proposition A.3. The following diagram commutes:



References

[1] Samson Abramsky and Nikos Tzevelekos. Introduction to categories and categorical logic. *New structures for physics*, pages 3–94, 2011.

[2] Henk P Barendregt. Introduction to lambda calculus. 1984.

[3] Timothy G Griffin. A formulae-as-type notion of control. In *Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 47–58, 1989.

[4] Joachim Lambek and Philip J Scott. *Introduction to higher-order categorical logic*, volume 7. Cambridge University Press, 1988.

$\sim$ These are seminar notes for Filippo Bonchi's course on *Category Theory and Cybernetic Culture*, presented by Lorenzo Pace in May 2024 at the CS Department of the University of Pisa, Italy.

Version 1.2, write to `l.pace4@studenti.unipi.it` for corrections, suggestions, insults and love letters.