Category Theory and the Simply-Typed λ -Calculus

Alfio Martini

Bericht-Nr. 96-7

Category Theory and the Simply Typed λ -Calculus

Alfio Martini

e-mail:alfio@cs.tu-berlin.de

Abstract

This report deals with the question on how to provide a categorical model for the simply-typed λ -calculus. We first introduce cartesian closed categories and work in detail a number of results concerning this construction. Next, we present the basic concepts related with the typed λ -calculus, i.e., concrete syntax for λ -terms, occurrence of variables, context substitution and equivalence of λ -terms. Then we present the typing rules and an equational proof system together with reduction rules that model the execution of expressions (programs). The chapter ends with the presentation of a categorical semantics for the calculus and a soundness proof for the equational proof system. The main technical result of this proof is the *substitution lemma*, which says, basically, that the (operational) concept of substitution can be understood (algebraically) as a composition of two suitable morphisms in a (cartesian closed) category.

Contents

1	Cartesian closed categories	2
	1.1 Exponentials	. 2
	1.2 Cartesian closed categories	. 4
2	The simply typed lambda calculus	17
	2.1 Basic concepts	. 17
	2.2 A categorical semantics for $\lambda^{\to,\times}$. 29
3	Bibliographical remarks	41
\mathbf{A}	Categorical Background	42
	A.1 Categories	. 42
	A.2 Initial and terminal objects	. 45
	A.3 Products	. 45
	A.4 Functors	. 46
	A.5 Natural transformations	. 49

1 Cartesian closed categories

In computer science, we may be interested in computing with *procedures* as arguments, i.e., we may need to describe higher type functions.

With "cartesian" categories, i.e., categories with all binary products and a terminal object, the notion of morphisms taking morphisms as arguments does not make sense. What we first need, is a further closure property, namely, the existence within the category, of an object B^A , which suitably represents the set of morphisms from A to B.

Especially, in this section, we first introduce the concepts of exponential object and of cartesian closed category. Next, we present some examples for these two constructions and state some elementary results, which will be used in the sequel so as to use cartesian closed categories to "interpret" the simply typed λ -calculus.

1.1 Exponentials

Given the sets A and B, we can form in **Set**, the set B^A , of all functions that have domain A and codomain B. In order to characterize B^A by morphisms, we observe that associated with B^A is a special morphism

$$eval_{AB}: B^A \times A \rightarrow B$$
,

given by the rule

$$eval_{A,B}(\langle f, a \rangle) = f(a),$$

i.e., the evaluation function. Its inputs are pairs of the form $\langle f, a \rangle$, where $f: A \to B$ and $a \in A$. The action of $eval_{A,B}$ for such input is to apply f to a or evaluate f at a, yielding the output $f(a) \in B$. The categorical description of B^A comes from the fact that $eval_{A,B}$ enjoys a universal property amongst all set functions of the form

$$C \times A \rightarrow B$$
.

Given any such g, there is one and only one function $curry_C(g): C \to B^A$ such that $eval_{A,B} \circ (curry_C(g) \times id_A) = g$.

The idea behind the definition of $curry_C(g)$ is that the action of g causes any particular c to determine a function $A \to B$ by fixing the first element of arguments of g at c, and allowing the second element to range over A. In other words, for a given $c \in C$ we define $g_c : A \to B$ by the rule

$$g_c(a) = g(\langle c, a \rangle)$$

for all $a \in A$. The function $curry_C(g): C \to B^A$ can now be defined by $curry_C(g)(c) = g_c$ for all $c \in C$. Thus, for any $\langle c, a \rangle \in C \times A$ we have

$$eval_{A,B}(\langle curry_C(g)(c), id_A(a) \rangle) = eval_{A,B}(\langle g_c, a \rangle) = g_c(a) = g(\langle c, a \rangle)$$

But from $eval_{A,B}(\langle curry_C(g)(c), a \rangle) = g(\langle c, a \rangle)$, we have that $curry_C(g)$ must be the function that for input a gives output $g(\langle c, a \rangle)$, i.e., $curry_C(g)(c)$ must be g_c as defined above.

By abstraction, we get the following

Definition 1.1.1 Let C be a category with all binary products and let A and B be objects of C. An object B^A is an **exponential** object if there is a morphism $eval_{A,B}: (B^A \times A) \to B$ such that for any object C and morphism $g: (C \times A) \to B$, there is a unique morphism $curry_C(g): C \to B^A$, such that $eval_{A,B} \circ (curry_C(g) \times id_A) = g$, i.e., such that the following diagram commutes:

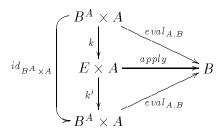
$$B \overset{eval_A,B}{\longleftrightarrow} A \times A$$

$$C \times A$$

$$C \times A$$

Proposition 1.1.2 Exponential objects are unique up to isomorphism.

Proof: Assume B^A and E are exponential objects and consider the following diagram, where $k = curry_{B^A}(eval_{A,B}) \times id_A$ and $k' = curry_E(apply) \times id_A$:



Note that the upper triangle commutes because E is an exponential object with morphism $apply: E \times A \to B$ while the lower triangle commutes because B^A is an exponential object with morphism $eval_{A,B}: B^A \times A \to B$. Thus the commutativity of the two triangles asserts the validity of the following equality:

$$eval_{A,B} = eval_{A,B} \circ (curry_E(apply) \times id_A) \circ (curry_{B^A}(eval_{A,B}) \times id_A)$$

However, we have

$$\begin{aligned} &(curry_E(apply) \times id_A) \circ (curry_{B^A}(eval_{A,B}) \times id_A) \\ &= (curry_E(apply) \circ curry_{B^A}(eval_{A,B})) \times (id_A \circ id_A) \\ &= (curry_E(apply) \circ curry_{B^A}(eval_{A,B}) \times id_A \end{aligned}$$
 (identity)

and thus

$$eval_{A,B} = eval_{A,B} \circ (curry_E(apply) \circ curry_{B^A}(eval_{A,B})) \times id_A$$

But note that $id_{B^A} \times id_A = id_{B^A \times A}$ (by A.3.7) and hence $eval_{A,B} = eval_{A,B} \circ id_{B^A \times A} = eval_{A,B} \circ (id_{B^A} \times id_A)$ (identity). This shows that both id_{B^A} and $curry_E(apply) \circ curry_{B^A}(eval_{A,B})$ are solutions to the equation

$$eval_{A,B} = eval_{A,B} \circ ((?) \times id_A)$$

Since B^A is an exponential object, we have, by uniqueness, that

$$curry_E(apply) \circ curry_{B^A}(eval_{A,B}) = id_{B^A}.$$

Interchanging the roles of B^A and E in the above argument leads to $curry_{B^A}(eval_{A,B}) \circ curry_E(apply) = id_E$.

Definition 1.1.3 A category that has a exponential object B^A for each pair of objects A and B is said to have exponentiation.

1.2 Cartesian closed categories

There are several possibilities for defining cartesian closed categories which differ on the amount of category theory that is necessary. The following is the simplest one:

Definition 1.2.1 A cartesian close category (CCC) is a category that has a terminal object 1, all binary products and for each pair of objects A and B an exponential object B^A .

Example 1.2.2 Set is a cartesian category and, as discussed before, has an exponential object B^A for each pair of objects A and B. Hence **Set** is a **CCC**.

Example 1.2.3 If A and B are finite sets, with say m and n elements, then B^A is finite and has n^m ("n to the power m) elements. In the expression n^m , the "m" is called an *exponent*, hence the above terminology.

Example 1.2.4 A **chain** is a poset (P, \leq) that is linearly ordered, i.e., has $p \leq q$ or $q \leq p$ for all $p, q \in P$. If (P, \leq) is a chain with a terminal object 1 (the maximal element), then we define

$$q^{p} = \begin{cases} 1 & \text{if } p \leq q \\ q & \text{if } q$$

A chain always has products:

$$q \times p = g.l.b(q,p) = \begin{cases} p & \text{if } p \leq q \\ q & \text{if } q < p. \end{cases}$$

We thus have two cases to consider for $eval_{pq}$:

- 1. $p \leq q$. Then $q^p \times p = 1 \times p = p \leq q$;
- 2. q < p. Then $q^p \times p = q \times p = q \le q$.

In either case $q^p \times p \leq q$ and so $eval_{pq}$ is the unique morphism $q^p \times q \rightarrow q$ in (P, \leq) (see the next diagram).

$$q \stackrel{eval_{pq}}{\longleftarrow} q^p \times p$$

$$q \stackrel{curry_C(g) \times id_p}{\longleftarrow} q \times p$$

Example 1.2.5 Let C(P) be the poset (P, \leq_P) considered as a category (see A.1.4). In this case, what means saying that C is a CCC? Think of the elements of P as "propositions" and interpret $p \leq q$ as "if p is assumed then q can be proved".

As shown in the example A.3.3, a product $p \times q$ coincides with the g.l.b of p and q. Thus we will write $p \wedge q$ for $p \times q$, and will interpret this as "p and q". To represent the exponential object q^p , we will write $p \Rightarrow q$, with the interpretation "p implies q". Now consider the following diagram:

$$q \overset{eval_{p,q}}{\longleftarrow} (p \Rightarrow q) \land p$$

$$r = \leq \qquad \uparrow^{curry_p(r) \times id_p}$$

$$r \land p$$

Then $eval_{p,q}:(p\Rightarrow q)\wedge p\to q$ is just the deduction rule known as modus ponens, for the logicians being the assertion

$$((p \Rightarrow q) \land p) \le q.$$

What it is really been told here, it is that, given a morphism $eval_{p,q}$ (i.e., the deduction rule modus ponens), r exists if and only if curry(r) exists, i.e., "from r and p we can deduce q if and only if from r we can deduce $p \Rightarrow q$ " for all $p, q, r \in (P, \leq)$.

Proposition 1.2.6 Let C be a CCC. Then for every C-morphism $eval_{A,B}: B^A \times A \to B$ we have that $curry_{B^A}(eval_{A,B}) = id_{B^A}$.

Proof: Consider the following diagram:

$$B \stackrel{eval_{A,B}}{\longleftarrow} B^A \times A$$

$$eval_{A,B} \qquad id_{BA} \times id_A$$

$$B^A \times A$$

This diagram is clearly commutative. To see this note that

$$eval_{A,B} \circ (id_{B^A} \times id_A) =$$
 $= eval_{A,B} \circ id_{B^A \times A}$ (proposition A.3.7)
 $= eval_{A,B}$ (identity)

But according to the definition of exponential object, $curry_{B^A}(eval_{A,B})$ is the only morphism which makes the diagram commute. So, by uniqueness, we have $curry_{B^A}(eval_{A,B}) = id_{B^A}$.

Proposition 1.2.7 For any morphism $h: C \to B^A$ we have that $h = curry_C(eval_{A,B} \circ (h \times id_A))$.

Proof: Consider the following diagram:

$$B \overset{eval_{A,B}}{\swarrow} B^A \times A$$

$$eval_{A,B} \circ (h \times id_A) \qquad h \times id_A$$

$$C \times A$$

It is clear that this diagram commutes. However, according to the definition of exponential object, $curry_C(eval_{A,B} \circ (h \times id_A))$ is the only morphism which makes the diagram commute. Thus, as h also does the job, we have, by uniqueness, that $curry_C(eval_{A,B} \circ (h \times id_A)) = h$.

Proposition 1.2.8 If C has exponentiation, then there is a bijection between the set of C-morphisms from $C \times A$ to B and the set of those from C to B^A , i.e., a bijective correspondence between $Hom_{\mathbf{C}}(C \times A, B)$ and $Hom_{\mathbf{C}}(C, B^A)$.

Proof: We have to show that, given $curry_C : Hom_C(C \times A, B) \to Hom_{\mathbf{C}}(C, B^A)$ and $curry_C^{-1} : Hom_{\mathbf{C}}(C, B^A) \to Hom_{\mathbf{C}}(C \times A, B)$, we have $curry_C^{-1} = id_{Hom_{\mathbf{C}}(C, B^A)} \land curry_C^{-1} \circ curry_C = id_{Hom_{\mathbf{C}}(C \times A, B)}$. Therefore, given $h \in Hom_{\mathbf{C}}(C, B^A)$ we define $curry_C^{-1}(h) = eval_{A,B} \circ (h \times id_A)$ (see the corresponding diagram in the previous proposition).

Now we have:

$$curry_C \circ curry_C^{-1}(h) = curry_C(curry_C^{-1}(h))$$
 (composition)
= $curry_C(eval_{A,B} \circ (h \times id_A))$ (by definition of $curry_C^{-1}$)
= h (by 1.2.7)

while for any $g \in Hom_{\mathbf{C}}(C \times A, B)$ we have:

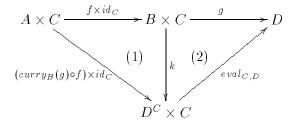
Propositions 1.2.7 and 1.2.8 allow us to state another equivalent definition of an exponential object:

Definition 1.2.9 Let C be a category with all binary products and let $A, B \in Ob(C)$. An exponential object of A and B is an object B^A together with a morphism $eval_{A,B}: B^A \times A \to B$ (evaluation map), and for every object C an operation $curry_C :: Hom_C(C \times A, B) \to Hom_C(C, B^A)$ such that for all morphisms $g: C \times A \to B, h: C \to B^A$, the following equations hold:

- 1. $eval_{A,B} \circ (curry_C(g) \times id_A) = g;$
- 2. $curry_C(eval_{A,B} \circ (h \times id_A)) = h$.

Proposition 1.2.10 As a consequence of the definition of exponential object we have that for any $f: A \to B$ and $g: B \times C \to D$, $curry_B(g) \circ f = curry_A(g \circ (f \times id_C))$.

Proof: Consider the following diagram, where $k = curry_B(q) \times id_C$:



To see that (1) really commutes note that

$$\begin{aligned} &(curry_B(g) \times id_C) \circ (f \times id_C) = \\ &= (curry_B(g) \circ f) \times (id_C \circ id_C) \\ &= (curry_B(g) \circ f) \times id_C \end{aligned}$$
 (proposition A.3.5)

Note that (2) commutes by definition of exponential object.

On the other hand, (1)+(2) also commutes, since that

$$\begin{aligned} eval_{C,D} \circ (curry_B(g) \circ f) \times id_C) &= \\ &= eval_{C,D} \circ ((curry_B(g) \times id_C) \circ (f \times id_C)) \\ &= (eval_{C,D} \circ (curry_B(g) \times id_C)) \circ (f \times id_C) \\ &= g \circ (f \times id_C) \end{aligned} \tag{by (1)}$$

Thus, we have shown that the following diagram is commutative:

$$A \times C \xrightarrow{g \circ (f \times id_C)} D$$

$$(curry_B(g) \circ f) \times id_C \downarrow \qquad eval_{C,D}$$

$$D^C \times C$$

Now, according to the definition of exponential object, $curry_A(g \circ (f \times id_C))$ is the only morphism which makes (1)+(2) commute. As $curry_B(g) \circ f$ also does the job, we must have, by uniqueness, $curry_B(g) \circ f = curry_A(g \circ (f \times id_C))$.

The following theorem can be ommitted on a first (and even on a second... reading).

Theorem 1.2.11 The following natural isomorphisms hold in all CCC's, for any object A, B, and C:

- 1. $A \cong A$;
- 2. $1 \times A \cong A$;
- 3. $A \times B \cong B \times A$;
- 4. $(A \times B) \times C \cong A \times (B \times C)$;
- 5. $C^{A \times B} \cong C^{B^A}$;
- 6. $(B \times C)^A \cong (B^A) \times (C^A)$;
- 7. $A^1 \cong A$;
- 8. $1^A \cong 1$.

Proof: We adopt the following approach in proving the above natural isomorphisms: for each one of them we provide the functors involved and define the components of the natural transformation in one of the directions. For the other direction, we define the corresponding family of inverses and show an isomorphism between these two families. Then naturality of the family of inverses follows directly from A.5.5. The functors necessary to prove these facts are introduced in the appendix. We leave to the reader to show that they are really defining functors or we refer to [15], where explicit calculations are given. Please note that, as stated in the theorem, these isomorphisms are "natural" in all the variables. This means: we fix all but one variable. Then both sides of the isomorphism are functors in the remaining variable, a fact which will become clear as we proceed in the calculations.

Proof of 1: This is trivial. The natural isomorphism holds between the two identity functors in \mathbf{C} . Therefore, the identity natural transformation $\iota: I_{\mathbf{C}} \to I_{\mathbf{C}}$ is defined for each component $A \in \mathbf{C}$ by $\iota(A) = id_A$ such that for each $f: A \to B$ in \mathbf{C} the following diagram commutes (ommitting the identity functor):

$$\begin{array}{ccc}
A & A & \xrightarrow{id_A} A \\
f \downarrow & f \downarrow & \downarrow f \\
B & B & \xrightarrow{id_B} B
\end{array}$$

For each $A \in \mathbf{C}$ we define the inverse by $\iota^{-1}(A) = id_A$ such that trivially $\iota_A \circ \iota_A^{-1} = id_A = \iota_A^{-1} \circ \iota_A$.

Proof of 2: Here the natural isomorphism holds between the left product functor $1 \times -: C \to C$ and the identity functor on \mathbb{C} . We define the natural transformation $\pi: 1 \times - \to I_{\mathbb{C}}$ at each component $A \in \mathbb{C}$ by the projection $\pi_A: 1 \times A \to A$. Now we have to show that the following diagram commutes for every $f: A \to B$ in \mathbb{C} .

$$\begin{array}{ccc}
A & 1 \times A \xrightarrow{\pi_A} A \\
f \downarrow & id_1 \times f \downarrow & \downarrow f \\
B & 1 \times B \xrightarrow{\pi_B} B
\end{array}$$

To see this, consider that

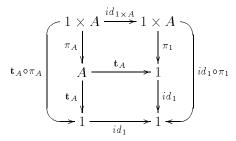
$$\pi_B \circ \langle id_1 \times f \rangle = \pi_B \circ \langle id_1 \circ \pi_1, f \circ \pi_A \rangle \qquad \text{(by A.3.2)} \\
= f \circ \pi_A \qquad \text{(by definition of } \pi_A)$$

This shows that $\pi: 1 \times - \to I_{\mathbf{C}}$ is a natural transformation. The inverse natural transformation $\pi^{-1}: I_{\mathbf{C}} \to 1 \times -$ is given at each component A by $\pi_A^{-1} = \langle \mathbf{t}_A, id_A \rangle$ which is well-defined by the universal property of the product $1 \times A$ in the following (commutative) diagram:

$$1 \xrightarrow[\mathbf{t}_{1} \times A]{\mathbf{t}_{A}, id_{A}} \xrightarrow{id_{A}} A$$

where $\mathbf{t}_?$ denotes the unique morphism from ? to the terminal object 1. Now, for each $A \in \mathbf{C}$ we have

$$\pi_A^{-1} \circ \pi_A = \langle \mathbf{t}_A, id_A \rangle \circ \pi_A$$
 (by definition of π^{-1})
$$= \langle \mathbf{t}_A \circ \pi_A, id_A \circ \pi_A \rangle$$
 (by A.3.6)
$$= \langle id_1 \circ \pi_1, id_A \circ \pi_A \rangle$$
 (1 is terminal, see diagram below)
$$= id_1 \times id_A$$
 (by A.3.2)
$$= id_{1 \times A}$$
 (by A.3.7)



On the other hand, we have:

$$\pi_A \circ \pi_A^{-1} = \pi_A \circ \langle \mathbf{t}_A, id_A \rangle$$
 (by definition of π^{-1})
= id_A (by definition of π_A)

This shows that at each component $A \in \mathbf{A}$, $\pi_A \circ \pi_A^{-1} = id_A \wedge \pi_A^{-1} \circ \pi_A = id_{1 \times A}$.

Proof of 3: Here the natural isomorphism holds between the right product functor $-\times B$: $\mathbf{C} \to \mathbf{C}$ and the left product functor $B \times -: \mathbf{C} \to \mathbf{C}$. We define the natural transformation $twist: -\times B \to B \times -$ at each component $A \in \mathbf{C}$ by $twist_A = \langle \pi_B, \pi_A \rangle$. To check that twist is natural requires to show that the following diagram commutes for each $f: A \to B$ in \mathbf{C} :

$$\begin{array}{ccc} A & A \times B \xrightarrow{twist_A} B \times A \\ f & & f \times id_B \\ B & & B \times B \xrightarrow{twist_B} B \times B \end{array}$$

To see this, note that

$$(id_B \times f) \circ twist_A = (id_B \times f) \circ \langle \pi_B, \pi_A \rangle \qquad \text{(by definition of } twist)$$

$$= \langle id_B \circ \pi_B, f \circ \pi_A \rangle \qquad \text{(by A.3.8.1)}$$

$$= \langle \pi_B, \pi_B \rangle \circ \langle f \circ \pi_A, id_B \circ \pi_B \rangle \qquad \text{(by definition of } \pi_B)$$

$$= twist_B \circ \langle f \circ \pi_A, id_B \circ \pi_B \rangle \qquad \text{(by definition of } twist)$$

$$= twist_B \circ (f \times id_B) \qquad \text{(by A.3.2)}$$

This shows that $twist : -\times B \to B \times -$ is a natural transformation. Now we define the natural transformation $twist^{-1} : B \times - \to - \times B$ at each component $A \in \mathbf{C}$ by $twist_A^{-1} = \langle \pi_A, \pi_B \rangle$. Now we have:

$$twist_A \circ twist_A^{-1} = \langle \pi_B, \pi_A \rangle \circ \langle \pi_A, \pi_B \rangle \quad \text{(by definition of } twist \text{ and } twist^{-1})$$

$$= \langle \pi_B, \pi_A \rangle \quad \text{(by definition of } \pi_A \text{ and } \pi_B)$$

$$= \langle id_B \circ \pi_B, id_A \circ \pi_A \rangle \quad \text{(identity)}$$

$$= id_B \times id_A \quad \text{(by A.3.2)}$$

$$= id_{B \times A} \quad \text{(by A.3.7)}$$

while

$$twist_A^{-1} \circ twist_A = \langle \pi_A, \pi_B \rangle \circ \langle \pi_B, \pi_A \rangle \quad \text{(by definition of } twist \text{ and } twist^{-1})$$

$$= \langle \pi_A, \pi_B \rangle \quad \text{(by definition of } \pi_A \text{ and } \pi_B)$$

$$= \langle id_A \circ \pi_A, id_B \circ \pi_B \rangle \quad \text{(identity)}$$

$$= id_A \times id_B \quad \text{(by A.3.2)}$$

$$= id_{A \times B} \quad \text{(by A.3.7)}$$

This shows that at each component $A \in \mathbb{C}$, $twist_A \circ twist_A^{-1} = id_{B \times A} \wedge twist_A^{-1} \circ twist_A = id_{A \times B}$.

Proof of 4: Here the natural isomorphism holds between the composite functor (see A.4.14) $(-\times C) \circ (A \times -) : \mathbf{C} \to \mathbf{C}$ (where $-\times C : \mathbf{C} \to \mathbf{C}$ and $A \times -: \mathbf{C} \to \mathbf{C}$) and the composite functor $(A \times -) \circ (-\times C) : \mathbf{C} \to \mathbf{C}$, which are defined for each $B \in \mathbf{C}$ and each $f : B \to D$ in \mathbf{C} by

$$(- \times C) \circ (A \times -)(B) = (- \times C)(A \times B) = (A \times B) \times C$$
$$(- \times C) \circ (A \times -)(f) = (- \times C)(id_A \times f) = (id_A \times f) \times id_C$$

while

$$(A \times -) \circ (- \times C)(B) = (A \times -)(B \times C) = A \times (B \times C)$$
$$(A \times -) \circ (- \times C)(f) = (A \times -)(f \times id_C) = id_A \times (f \times id_C)$$

Now we define the natural transformation $\alpha: (-\times C) \circ (A \times -) \xrightarrow{\cdot} (A \times -) \circ (-\times C)$ at each component $B \in \mathbf{C}$ by

$$\alpha_B = \langle \pi_A \circ \pi_{A \times B}, \langle \pi_B \circ \pi_{A \times B}, \pi_C \rangle \rangle$$

Now we have to show that, for each $f: B \to D$ in C, the following diagram commutes:

$$B \qquad (A \times B) \times C \xrightarrow{\alpha_B} A \times (B \times C)$$

$$f \downarrow \qquad (id_A \times f) \times id_C \qquad id_A \times (f \times id_C)$$

$$D \qquad (A \times D) \times C \xrightarrow{\alpha_D} A \times (D \times C)$$

To see this, note that, by chasing the above diagram counterclockwise we have:

```
\begin{split} &\alpha_D \circ (id_A \circ f) \circ id_C = \\ &= \langle \pi_A \circ \pi_{A \times D}, \langle \pi_D \circ \pi_{A \times D}, \pi_C \rangle \rangle \circ (id_A \circ f) \circ id_C \\ &\text{(by definition of } \alpha) \\ &= \langle \pi_A \circ \pi_{A \times D}, \langle \pi_D \circ \pi_{A \times D}, \pi_C \rangle \rangle \circ \langle (id_A \circ f) \circ \pi_{A \times B}, id_C \circ \pi_C \rangle \\ &\text{(by A.3.2)} \\ &= \langle \pi_A \circ \pi_{A \times D}, \langle \pi_D \circ \pi_{A \times D}, \pi_C \rangle \rangle \circ \langle \langle id_A \circ \pi_A, f \circ \pi_B \rangle \circ \pi_{A \times B}, id_C \circ \pi_C \rangle \\ &\text{(by A.3.2)} \\ &= \langle \pi_A \circ \pi_{A \times D}, \langle \pi_D \circ \pi_{A \times D}, \pi_C \rangle \rangle \circ \langle \langle id_A \circ \pi_A \circ \pi_{A \times B}, f \circ \pi_B \circ \pi_{A \times B} \rangle, id_C \circ \pi_C \rangle \\ &\text{(by A.3.6)} \\ &= \langle \pi_A \circ \langle id_A \circ \pi_A \circ \pi_{A \times B}, f \circ \pi_B \circ \pi_{A \times B} \rangle, \langle \pi_D \circ \langle id_A \circ \pi_A \circ \pi_{A \times B}, f \circ \pi_B \circ \pi_{A \times B} \rangle, id_C \circ \pi_C \rangle \\ &\text{(by definition of } \pi_{A \times D}, \pi_C) \\ &= \langle id_A \circ \pi_A \circ \pi_{A \times B}, \langle f \circ \pi_B \circ \pi_{A \times B}, id_C \circ \pi_C \rangle \rangle \\ &\text{(by definition of } \pi_A, \pi_D) \\ &= \langle \pi_A \circ \pi_{A \times B}, \langle f \circ \pi_B \circ \pi_{A \times B}, \pi_C \rangle \rangle \\ &\text{(identity)} \end{split}
```

while chasing it clockwise it holds that

```
\begin{split} &id_A \times (f \times id_C) \circ \alpha_B = \\ &= id_A \times (f \times id_C) \circ \langle \pi_A \circ \pi_{A \times B}, \langle \pi_B \circ \pi_{A \times B}, \pi_C \rangle \rangle \\ &(\text{by definition of } \alpha) \\ &= \langle id_A \circ \pi_A, (f \times id_C) \circ \pi_{B \times C} \rangle \circ \langle \pi_A \circ \pi_{A \times B}, \langle \pi_B \circ \pi_{A \times B}, \pi_C \rangle \rangle \\ &(\text{by A.3.2}) \\ &= \langle id_A \circ \pi_A, \langle f \circ \pi_B, id_C \circ \pi_C \rangle \circ \pi_{B \times C} \rangle \circ \langle \pi_A \circ \pi_{A \times B}, \langle \pi_B \circ \pi_{A \times B}, \pi_C \rangle \rangle \\ &(\text{by A.3.2}) \\ &= \langle id_A \circ \pi_A, \langle f \circ \pi_B \circ \pi_{B \times C}, id_C \circ \pi_C \circ \pi_{B \times C} \rangle \rangle \circ \langle \pi_A \circ \pi_{A \times B}, \langle \pi_B \circ \pi_{A \times B}, \pi_C \rangle \rangle \\ &(\text{by A.3.6}) \\ &= \langle id_A \circ \pi_A \circ \pi_{A \times B}, \langle f \circ \pi_B \circ \langle \pi_B \circ \pi_{A \times B}, id_C \circ \pi_C \rangle, id_C \circ \pi_C \circ \langle \pi_B \circ \pi_{A \times B}, \pi_C \rangle \rangle \\ &(\text{by definition of } \pi_A, \pi_{B \times C}) \\ &= \langle id_A \circ \pi_A \circ \pi_{A \times B}, \langle f \circ \pi_B \circ \pi_{A \times B}, id_C \circ \pi_C \rangle \rangle \\ &(\text{by definition of } \pi_B, \pi_C) \\ &= \langle \pi_A \circ \pi_{A \times B}, \langle f \circ \pi_B \circ \pi_{A \times B}, \pi_C \rangle \rangle \\ &(\text{identity}) \end{split}
```

This shows that $\alpha: (-\times C) \circ (A \times -) \to (A \times -) \circ (-\times C)$ is a natural transformation. Now we define the inverse natural transformation $\alpha^{-1}: (A \times -) \circ (-\times C) \to : (-\times C) \circ (A \times -)$ at each component $B \in \mathbf{C}$ by

$$\alpha_B^{-1} = \langle \langle \pi_A, \pi_B \circ \pi_{B \times C} \rangle, \pi_C \circ \pi_{B \times C} \rangle$$

and we have:

$$\alpha_{B} \circ \alpha_{B}^{-1} = \langle \pi_{A} \circ \pi_{A \times B}, \langle \pi_{B} \circ \pi_{A \times B}, \pi_{C} \rangle \rangle \circ \langle \langle \pi_{A}, \pi_{B} \circ \pi_{B \times C} \rangle, \pi_{C} \circ \pi_{B \times C} \rangle$$
(by definition of α and α^{-1})
$$= \langle \pi_{A} \circ \langle \pi_{A}, \pi_{B} \circ \pi_{B \times C} \rangle, \langle \pi_{B} \circ \langle \pi_{A}, \pi_{B} \circ \pi_{B \times C} \rangle, \pi_{C} \circ \pi_{B \times C} \rangle \rangle$$
(by definition of $\pi_{A \times B}, \pi_{C}$)
$$= \langle \pi_{A}, \langle \pi_{B} \circ \pi_{B \times C}, \pi_{C} \times \pi_{B \times C} \rangle \rangle$$
(by definition of π_{A}, π_{B})
$$= \langle id_{A} \circ \pi_{A}, \langle id_{B} \circ \pi_{B} \circ \pi_{B \times C}, id_{C} \circ \pi_{C} \times \pi_{B \times C} \rangle \rangle$$
(identity)
$$= id_{A} \times ((id_{B} \circ \pi_{B}) \times (id_{C} \circ \pi_{C}))$$
(by A.3.2)
$$= id_{A} \times ((id_{B} \times id_{C}) \circ (\pi_{B} \times \pi_{C}))$$
(identity)
$$= id_{A} \times ((id_{B} \times id_{C}) \circ (id_{B} \circ \pi_{B}) \times (id_{C} \circ \pi_{C})))$$
(identity)
$$= id_{A} \times ((id_{B} \times id_{C}) \circ (id_{B} \times id_{C}))$$
(by A.3.2)
$$= id_{A} \times ((id_{B} \otimes id_{B}) \times (id_{C} \circ id_{C}))$$
(by A.3.5)
$$= id_{A} \times ((id_{B} \otimes id_{B}) \times (id_{C} \circ id_{C}))$$
(by A.3.5)
$$= id_{A} \times ((id_{B} \times id_{C}) \circ (id_{B} \times id_{C}))$$
(identity)
$$= id_{A} \times ((id_{B} \times id_{C}) \circ (id_{C} \circ id_{C}))$$
(identity)
$$= id_{A} \times ((id_{B} \times id_{C}) \circ (id_{C} \circ id_{C}))$$
(identity)
$$= id_{A} \times ((id_{B} \times id_{C}) \circ (id_{C} \circ id_{C}))$$
(identity)
$$= id_{A} \times ((id_{B} \times id_{C}) \circ (id_{C} \circ id_{C}))$$
(identity)

In a similar way it can be shown that $\alpha_B^{-1} \circ \alpha_B = id_{(A \times B) \times C}$.

Proof of 5: Here the natural isomorphism holds between the composite functor $Hom_{\mathbf{C}}(-,C) \circ (-\times B)^{op}: \mathbf{C^{op}} \to \mathbf{Set}$ (where $(-\times B)^{op}: \mathbf{C^{op}} \to \mathbf{C^{op}}$, and $Hom_{\mathbf{C}}(-,C): \mathbf{C^{op}} \to \mathbf{Set}$) and the hom-functor $Hom_{\mathbf{C}}(-,C^B): \mathbf{C^{op}} \to \mathbf{Set}$, which are defined for each $A \in \mathbf{C}$ and each $f: A \to D$ in \mathbf{C} (hence $f^{op}: D \to A$ in $\mathbf{C^{op}}$) by

$$Hom_{\mathbf{C}}(-,C) \circ (-\times B)^{op}(A) = Hom_{\mathbf{C}}(-,C)(A\times B) = Hom_{\mathbf{C}}(A\times B,C)$$

$$Hom_{\mathbf{C}}(-,C) \circ (-\times B)(f^{op}) = Hom_{\mathbf{C}}(-,C)(f^{op}\times id_B^{op}) = Hom_{\mathbf{C}}(f^{op}\times id_B^{op},C)$$

where $Hom_{\mathbf{C}}(f^{op} \times id_B^{op}, C) : Hom_{\mathbf{C}}(D \times B, C) \to Hom_{\mathbf{C}}(A \times B, C)$. Besides,

$$\begin{array}{l} \operatorname{Hom}_{\mathbf{C}}(-,C^B)(A) = \operatorname{Hom}_{\mathbf{C}}(A,C^B) \\ \operatorname{Hom}_{\mathbf{C}}(-,C^B)(f^{op}) = \operatorname{Hom}_{\mathbf{C}}(f^{op},C^B) : \operatorname{Hom}_{\mathbf{C}}(D,C^B) \to \operatorname{Hom}_{\mathbf{C}}(A,C^B) \end{array}$$

We define a natural transformation $curry: Hom_{\mathbf{C}}(-,C) \circ (-\times B) \to Hom_{\mathbf{C}}(-,C^B)$ at each component $A \in \mathbf{C}$ and for each $g: A \times B \to C$ by the map $g \mapsto curry_A(g)$. To check that this definition indeed delivers a natural transformation, it is necessary to show that the following diagram commutes for each $f: A \to D$ in \mathbf{C} (hence $f^{op}: D \to A$):

$$\begin{array}{ccc}
D & Hom_{\mathbf{C}}(D \times B, C) \xrightarrow{curry_{D}} Hom_{\mathbf{C}}(D, C^{B}) \\
\downarrow & & \downarrow & \\
Hom_{\mathbf{C}}(f^{op} \times id_{B}^{op}, C) & Hom_{\mathbf{C}}(f^{op}, C^{B}) \\
A & Hom_{\mathbf{C}}(A \times B, C)_{curry_{A}} Hom_{\mathbf{C}}(A, C^{B})
\end{array}$$

To see this, take any $g \in Hom_{\mathbf{C}}(D \times B, C)$. Chasing clockwise we have:

$$Hom_{\mathbf{C}}(f^{op}, C^B)(curry_D(g)) = curry_D(g) \circ f$$
 (by definition of $Hom_{\mathbf{C}}(f^{op}, C^B)$)

while in the other direction we have:

$$curry_A(Hom_{\mathbf{C}}(f^{op} \times id_B^{op}, C)(g)) = curry_A(g \circ (f \times id_B))$$

(by definition of $Hom_{\mathbf{C}}(f^{op} \times id_B^{op}, C))$

But now, the equality $curry_D(g) \circ f = curry_A(g \circ (f \times id_B))$ holds directly by proposition 1.2.10. This shows that $curry : Hom_{\mathbf{C}}(-,C) \circ (-\times B) \to Hom_{\mathbf{C}}(-,C^B)$ is a natural transformation.

The definition of the natural transformation $curry^{-1}: Hom_{\mathbf{C}}(-, C^B) \to Hom_{\mathbf{C}}(-, C) \circ (-\times B)$ at each component $A \in \mathbf{A}$ as well as the verification of the equations $curry_A \circ curry_A^{-1} = id_{Hom_{\mathbf{C}}(A,C^B)} \wedge curry_A^{-1} \circ curry_A = id_{Hom_{\mathbf{C}}(A\times B,C)}$ were done already in the proof of 1.2.8.

Proof of 6: Here the natural isomorphism holds between the composite functor

$$Hom_{\mathbf{C}\times\mathbf{C}}(-,\langle A,B\rangle)\circ\Delta^{op}:\mathbf{C^{op}}\to\mathbf{Set}$$

(where $\Delta^{op}: \mathbf{C^{op}} \to \mathbf{C^{op}} \times \mathbf{C^{op}}$ is the diagonal functor and $Hom_{\mathbf{C} \times \mathbf{C}}(-, \langle A, B \rangle): \mathbf{C^{op}} \times \mathbf{C^{op}} \to \mathbf{Set}$) and the hom-functor $Hom_{\mathbf{C}}(-, A \times B): \mathbf{C^{op}} \to \mathbf{Set}$, which are defined for each $C \in \mathbf{C}$ and for each $f: C' \to C$ in \mathbf{C} (hence $f^{op}: C \to C'$ in $\mathbf{C^{op}}$) by:

$$Hom_{\mathbf{C}\times\mathbf{C}}(-,\langle A,B\rangle) \circ \Delta^{op}(C) = Hom_{\mathbf{C}\times\mathbf{C}}(-,\langle A,B\rangle)(\langle C,C\rangle) = Hom_{\mathbf{C}\times\mathbf{C}}(\langle C,C\rangle,\langle A,B\rangle)$$

 $Hom_{\mathbf{C}\times\mathbf{C}}(-,\langle A,B\rangle) \circ \Delta^{op}(f^{op}) = Hom_{\mathbf{C}\times\mathbf{C}}(\langle f^{op},f^{op}\rangle,\langle A,B\rangle)$

where

$$Hom_{\mathbf{C}\times\mathbf{C}}(\langle f^{op}, f^{op}\rangle, \langle A, B\rangle) : Hom_{\mathbf{C}\times\mathbf{C}}(\langle C, C\rangle, \langle A, B\rangle) \to Hom_{\mathbf{C}\times\mathbf{C}}(\langle C', C'\rangle, \langle A, B\rangle)$$

$$Hom_{\mathbf{C}}(-, A \times B)(C) = Hom_{\mathbf{C}}(C, A \times B)$$

 $Hom_{\mathbf{C}}(-, A \times B)(f^{op}) = Hom_{C}(f^{op}, A \times B) : Hom_{\mathbf{C}}(C, A \times B) \to Hom_{\mathbf{C}}(C', A \times B)$

Now we define a natural transformation $\tau: Hom_{\mathbf{C}\times\mathbf{C}}(-,\langle A,B\rangle) \circ \Delta^{op} \to Hom_{\mathbf{C}}(-,A\times B)$ at each component $C \in \mathbf{C}$ and for each pair of morphisms $\langle g,k\rangle$ in $\mathbf{C}\times\mathbf{C}$ $(g:C\to A,k:C\to B)$ by

$$\tau_C(\langle g, k \rangle) = \langle g, k \rangle_C$$

To see that τ is indeed a natural transformation requires to show that the following diagram commutes for each $f: C' \to C$ in \mathbf{C} :

$$\begin{array}{c|c} C & Hom_{\mathbf{C}\times\mathbf{C}}(\langle C,C\rangle,\langle A,B\rangle) \xrightarrow{\tau_{C}} Hom_{\mathbf{C}}(C,A\times B) \\ \downarrow^{f^{op}} & Hom_{\mathbf{C}\times\mathbf{C}}(\langle f^{op},f^{op}\rangle,\langle A,B\rangle) & Hom_{\mathbf{C}}(f^{op},A\times B) \\ C' & Hom_{\mathbf{C}\times\mathbf{C}}(\langle C',C'\rangle,\langle A,B\rangle) \xrightarrow{\tau_{C'}} Hom_{\mathbf{C}}(C',A\times B) \end{array}$$

To see this, take any $\langle g: C \to A, k: C \to B \rangle \in \mathbf{C} \times \mathbf{C}$. Chasing clockwise we have:

$$Hom_{\mathbf{C}}(f^{op}, A \times B) \circ \tau_{C}(\langle g, k \rangle) = Hom_{\mathbf{C}}(f^{op}, A \times B)(\langle g, k \rangle_{C})$$

(by definition of τ)
$$= \langle g, k \rangle_{C} \circ f$$
(by definition of $Hom_{\mathbf{C}}(f^{op}, A \times B)$)
$$= \langle g \circ f, k \circ f \rangle_{C'}$$
(by A.3.6)

while counterclockwise we have:

$$\tau_{C'} \circ Hom_{\mathbf{C} \times \mathbf{C}}(\langle f^{op}, f^{op} \rangle, \langle A, B \rangle)(\langle g, k \rangle) = \tau_{C'}(\langle g, k \rangle \circ \langle f, f \rangle)$$
(by definition of $Hom_{\mathbf{C} \times \mathbf{C}}(\langle f, f \rangle, \langle A, B \rangle))$

$$= \tau_{C'}(\langle g \circ f, k \circ f \rangle)$$
(composition)
$$= \langle g \circ f, k \circ f \rangle_{C'}$$
(by definition of τ)

This shows that $\tau: Hom_{\mathbf{C}\times\mathbf{C}}(-, \langle A, B \rangle) \circ \Delta \to Hom_{\mathbf{C}}(-, A \times B)$ is a natural transformation.

We define now the inverse natural transformation

$$\tau^{-1}: Hom_{\mathbf{C}}(-, A \times B) \xrightarrow{\cdot} \tau: Hom_{\mathbf{C} \times \mathbf{C}}(-, \langle A, B \rangle) \circ \Delta^{op}$$

at each component $C \in \mathbf{C}$ for each $h: C \to B$ by

$$\tau_C^{-1}(h) = \langle \pi_A \circ h, \pi_B \circ h \rangle$$

Now we have:

$$\begin{array}{lll} \tau_C^{-1} \circ \tau_C(\langle g, k \rangle) & = & \tau_C^{-1}(\langle k, g \rangle_C) & \text{(by definition of } \tau) \\ & = & \langle \pi_A \circ \langle g, k \rangle_C, \pi_B \circ \langle g, k \rangle_C \rangle & \text{(by definition of } \tau^{-1}) \\ & = & \langle g, k \rangle & \text{(by A.3.4)} \end{array}$$

while

$$\tau_C \circ \tau_C^{1-}(h) = \tau_C(\langle \pi_A \circ h, \pi_B \circ h \rangle) \text{ (by definition of } \tau^{-1}) \\
= \langle \pi_A \circ h, \pi_B \circ h \rangle_C \text{ (by definition of } \tau) \\
= h \text{ (by A.3.4)}$$

We leave the for the reader the easy proof of 7 and go directly to

Proof of 8: Here, the natural isomorphism holds between the functors $1^{(-)}: \mathbb{C}^{op} \to \mathbb{C}$ and the constant functor $\Delta(1): \mathbb{C}^{op} \to \mathbb{C}$. The latter maps all objects of \mathbb{C}^{op} to the terminal object 1, and all the morphisms of \mathbb{C} to the identity $id_1: 1 \to 1$. The functor $1^{(-)}: \mathbb{C}^{op} \to \mathbb{C}$ is defined in the following way:

- For each object $A, 1^{(A)} = 1^A$;
- $\bullet \text{ for each morphism } f^{op}: B \rightarrow A, 1^{f^{op}}: 1^B \rightarrow 1^A = curry_{1^B}(eval_{B,1} \circ (id_{1^B} \times f)), \text{ where } f^{op}: B \rightarrow A, 1^{f^{op}}: 1^B \rightarrow 1^A = curry_{1^B}(eval_{B,1} \circ (id_{1^B} \times f)), \text{ where } f^{op}: B \rightarrow A, 1^{f^{op}}: 1^B \rightarrow 1^A = curry_{1^B}(eval_{B,1} \circ (id_{1^B} \times f)), \text{ where } f^{op}: A \rightarrow A, 1^{f^{op}}: 1^B \rightarrow 1^A = curry_{1^B}(eval_{B,1} \circ (id_{1^B} \times f)), \text{ of } f^{op}: A \rightarrow A, 1^{f^{op}}: A \rightarrow A, A$

$$1^B \times A \xrightarrow{id_{1B} \times f} 1^B \times B \xrightarrow{eval_{B,1}} 1$$

We define the natural transformation $\beta:1^{(-)}\to\Delta(1)$ at each component $A\in\mathbf{C^{op}}$ by $\beta_A=\mathbf{t}_{1^A}:1^A\to 1$, such that for each $f^{op}:A\to B$, the following diagram commutes because 1 is a terminal object.

$$A \qquad 1^{A} \xrightarrow{\mathbf{t}_{1}^{A}} 1 = \Delta(A)$$

$$f^{op} \downarrow \qquad \downarrow_{id_{1}}$$

$$B \qquad 1^{B} \xrightarrow{\mathbf{t}_{1}^{B}} 1 = \Delta(B)$$

Now, for each $A \in \mathbf{C^{op}}$, we define the inverse natural transformation β^{-1} at each component A by $\beta_A^{-1} = curry_1(\mathbf{t}_{1\times 1} \circ (id_1 \times \mathbf{t}_A)) : 1 \to 1^A$. Then we have:

$$\beta_A \circ \beta_A^{-1} = \mathbf{t}_{1^A} \circ curry_1(\mathbf{t}_{1\times 1} \circ (id_1 \times \mathbf{t}_A))$$
 (by definition)
= id_1 ($|Hom_{\mathbf{C}}(1,1)| = 1$)

while

$$\beta_A^{-1} \circ \beta_A = curry_1 \circ (\mathbf{t}_{1\times 1} \circ (id_1 \times A)) \circ \mathbf{t}_{1^A} \qquad \text{(by definition)}$$

$$= curry_1(\mathbf{t}_{1\times A}) \circ \mathbf{t}_{1^A} \qquad (|Hom_{\mathbf{C}}(1\times A, 1)| = 1)$$

$$= curry_{1^A}(\mathbf{t}_{1\times A} \circ (\mathbf{t}_{1^A} \times id_A)) \qquad (|Hom_{\mathbf{C}}(1^A \times A, 1)| = 1)$$

$$= id_{1^A} \qquad (1.2.6)$$

Remark 1.2.12 What is worth mentioning is that the above isomorphisms are exactly the isomorphisms that hold in all CCC's, i.e., no other isomorphism is valid in all CCC's. The proof of this fact is nontrivial and is a nice application of the lambda calculus to categories.

2 The simply typed lambda calculus

In the following we start with the presentation of the basic concepts related with the typed λ -calculus, i.e., concrete syntax for λ -terms, occurrence of variables, context substitution and equivalence of λ -terms. Then we present the typing rules and an equational proof system together with reduction rules that model the execution of expressions (programs). The chapter ends with the presentation of a categorical semantics for the calculus and a consistency proof for the equational proof system. The main result of this proof is the *substitution lemma*, which says, basically, that the (operational) concept of substitution can be understood (algebraically) as a composition of two suitable morphisms in a cartesian closed category.

2.1 Basic concepts

Most theoretical studies about types in programming languages are based on some type of typed λ -calculus, which can be considered both as a simple programming language and as a metalanguage for the comprehension of others (e.g., imperative languages). In general, a typed λ -calculus consists of a syntax for expressions, an equational proof system, a set of reduction rules which allow us to simplify or "execute" expressions, and some form of semantics.

Although the typed λ -calculus does not reflect all the idionsycrasies of programming languages, it is very useful in the study of many fundamental questions. For instance, the typed λ -calculus can provide "models" for programming languages much the same way that

Turing machines provide models of computation in machines. Also, when considerations about typing are relevant, it is natural to use a calculus which can reflect the type structure of the programming language. For additional information between the relation of programming languages and some form of typed λ -calculus, we refer to ([12], [7]).

The basic typed calculus of λ - abstraction and application is called **typed** λ -calculus with function types. This system is normally referred as λ^{\rightarrow} . The calculus λ^{\rightarrow} can be extended by adding cartesian products of types, together with the associated operation of pair construction $\langle \cdot, \cdot \rangle$ and the projection functions **fst** and **snd**. The resulting system, which we denote by $\lambda^{\rightarrow,\times}$, is a typed calculus with function and product types. In this report we will work with the system $\lambda^{\rightarrow,\times}$ with constants and we will denote it also by $\lambda^{\rightarrow,\times}$.

We shall define the notion of a signature for a functional type theory, which consists of basic data from which to build types and terms

Definition 2.1.1

- Let Σ₁ be a collection of ground types and form the types over Σ₁ by the syntax
 t ::= o | t⇒t | t × t
 where o ∈ Σ₁.
- 2. Let Σ_0 be a set of pairs (c,t) where c is a term constant (or function symbol) and t is a type over Σ_1 (we assume that Σ_0 is a function). Form the language of terms over Σ_0 by the syntax

```
x \in Var
M ::= c \mid x \mid \lambda x : t.M \mid (MM) \mid (M,M) \mid \mathbf{fst}(M) \mid \mathbf{snd}(M).
where Var is a primitive syntactic set of variables.
```

These are the terms of $\lambda^{\to,\times}$ over the signature $\Sigma = (\Sigma_0, \Sigma_1)$.

Remarks 2.1.2

- 1. More clearly, Σ_0 is a set of function symbols each of which has an *arity* which is a natural number (possibly zero) and a *sorting*. For example, in the case that k is a function symbol of arity 0 we shall denote the sorting by k:t and the function symbol k will be referred to as a *constant* of type t.
- 2. Our $\lambda^{\to,\times}$ calculus with constants is also called $\lambda^{\to,\times}$ calculus with simple types, since it does not allow the possibility of having variables in the syntactic set of types.
- 3. We use letters from the end of the alphabet, such as x, y and z to range over variables. Types are normally written using letters r, s and t. Terms are normally written with letters L, M and N.
- 4. A type $\mathbf{o} \in \Sigma_1$ is called **ground** (or basic), and types $s \Rightarrow t$ and $s \times t$ are called **composite**.
- 5. Terms of the form $\lambda x : t.M$ are called **abstractions**, and terms of the form (MN) are called **applications**. Sometimes, we can use the notation M(N), to mimic the mathematical notation f(x) for an function applied to an argument.

6. The application operations associate to the left. Thus an application LMN should be parsed as (LM)N. Moreover application binds more tightly than abstraction, i.e., an expression $\lambda x: s.\lambda y: t.MN$ should be parsed as $\lambda x: s.\lambda y: t.(MN)$.

Definition 2.1.3 An **occurrence** of a variable in an expression is simply a point in the expression where the variable appears. For instance, the underscore indicates:

- 1. the first occurrence of x in $\lambda \underline{x}$: t.(yx)x;
- 2. the second occurrence of x in $\lambda x : t.(y\underline{x})x$;
- 3. the third occurrence of $x \ \lambda x : t.(yx)x$.

Definition 2.1.4 In an abstraction $\lambda x:t.M$, the first occurrence of x is called the **binding** occurrence of the abstraction. The variable x is called the **formal parameter**. The term M is called body of the abstraction, and t the **type tag** of the binding occurrence. In an application MN, the term M is the **operator** and N the **operator** or **argument**.

Definition 2.1.5 Given a term M, the set FV(M), of free variables of M is defined as follows:

- if M is x, then $FV(M) = \{x\}$;
- if M is $\lambda x : t.N$, then $FV(M) = FV(N) \{x\}$;
- if M is LN, then $FV(M) = FV(N) \cup FV(L)$;
- if M is (N, L), then $FV(M) = FV(N) \cup FV(L)$;
- if M is fst(N), then FV(M) = FV(N);
- if M is $\operatorname{snd}(N)$, then FV(M) = FV(N).

Let M be a term and suppose we are given a binding occurrence of a variable x in M. Suppose, in particular, that the binding occurrence of x in question is in a subterm $\lambda x:t.N$ of M. Then:

Definition 2.1.6 An occurrence of x in $\lambda x.N$ is in the **scope** of this binding if it is in FV(N). Free occurrences of x in N are said to be **bound** to the binding occurrence of x in the abstraction.

Definition 2.1.7 The context substitution [M/x]N of a term M for a variable x in a term N is inductively defined as follows:

```
if N is x:, then [M/x]N is M;
```

if N is y, $x \neq y$, then [M/x]N is N;

if N is $\lambda x : t.L$, then [M/x]N is $\lambda x : t.L$;

```
if N is \lambda y: t.L, y \notin FV(M), then [M/x]N is \lambda y: t.[M/x]L;
if N is N_1N_2, then [M/x]N is [M/x]N_1[M/x]N_2;
if N is (N_1, N_2), then [M/x]N is ([M/x]N_1, [M/x]N_2);
if N is \mathbf{fst}(L) then [M/x]N is \mathbf{fst}([M/x]L);
if N is \mathbf{snd}(L) then [M/x]N is \mathbf{snd}([M/x]L).
```

We shall say that a term M is α -equivalent to M' if they differ only in their bound variables. For example, the terms $\lambda x:t.(x,y)$ and $\lambda z.t.(z,y)$ are α equivalent. As another example, note that in the term $(\lambda x:t.x)x$ the variable x is both free and bound. This term is α -equivalent to, for example, $(\lambda y:t.y)x$. Any term M clearly determines an α -equivalent class for which M is representative. We shall also refer to such an equivalence class as a "term". From now on, "a term M" will mean the α -equivalence class determined by M. Note that we could refer to the notion of α -equivalence class as the "the renaming of bound variables". This discussion is formalized in the following

Definition 2.1.8 The α -equivalence of terms is defined to be the least relation \equiv between terms such that

```
c \equiv c x \equiv x; MN \equiv M'N' \text{ if } M \equiv M' \text{ and } N \equiv N'; \lambda x : t.M \equiv \lambda y : t.N \text{ if } [z/x]M \equiv [z/y]N, \text{ where } z \text{ is a variable that has no occurrence in } M \text{ or } N. (M,N) \equiv (M',N') \text{ if } M \equiv M' \text{ and } N \equiv N'. \mathbf{fst}(M) \equiv \mathbf{fst}(M') \text{ if } M \equiv M'. \mathbf{snd}(M) \equiv \mathbf{snd}(M') \text{ if } M \equiv M'.
```

Proposition 2.1.9 The relation \equiv defined in 2.1.7 is an equivalence relation.

Remarks 2.1.10

1. The equivalence classes of terms modulo the relation \equiv are called λ -terms, or more simply, terms. The equivalence class of terms determined by a term M is called its α -equivalence class. In general, if L is a term, then we mean that L is representative of its α -equivalence class.

2. When a term representative of an α -equivalence class is chosen, the name of the bound variable of the representative is distinct from the names of free variables in other terms being discussed.

3. We write $s \equiv t$ to indicate that the types s and t generated from our grammar for types are the same.

Programming languages are often defined with the use of some context free grammar, as in definition 2.1.1. However, a context free description of a statically typed language tells only half of the story, since that type restrictions are typically context sensitive. For instance, we would generally consider x + 3 well typed only if x is a numeric variable. In most of the programming languages, the declaration of x can precede the expression x + 3 by an arbitrary number of symbols and hence, there can be no free context grammar which can precisely generate the well typed terms (according to [11]).

Here, we are going to define a typed language using a formalism based on logic. Specifically, we are going to define expressions and their types simultaneously, using axioms and inference rules. Moreover, in order to capture the context sensitive syntax of the expressions which declare the type of the variables, we will have to to use type assignments together with the axioms and rules of inference. The formal definitions are presented in the sequel.

Definition 2.1.11 A type assignment is a list $H \equiv x_1 : t_1, \dots, x_n : t_n$ of pairs of variables and types, such that the variables x_i are distinct.

Remark 2.1.12 The empty type assignment is the degenerate case in which there are no pairs. We write $x:t\in H$ if x is x_i and t is t_i for some i. In this case it is said that x occurs in H, and this may abbreviated by writing $x\in H$. If $x:t\in H$, then we define H(x) as the type of t.

Definition 2.1.13 A typing judgement is a triple consisting of an assignment H, a term M and a type t, such that FV(M) appear in H. We write such triple in the form $H \vdash M : t$ and read it as "in the assignment H, the term M has type t".

Remark 2.1.14 Note that H can be seen as a "symbol table" which associates types with variables that may occur free in M.

There are two systems of rules describing the simply-typed λ -calculus. The first of these determines which of the terms which are generated by our grammar are to be viewed as well-typed. These are the terms to which we will assign a meaning in our semantic model. The second set of rules of the calculus forms its equational theory which consists of a set of rules for proving equalities between λ -terms.

The following set of axioms and rules determine our well-typed λ -terms.

[Constant]

$$\frac{c \in \Sigma_0}{H \vdash c : \Sigma_0(c)}$$

[Proj]

$$H', x:t, H'' \vdash x:t$$

[Abs]

$$\frac{H', x : s \vdash M : t}{H \vdash \lambda x : s.M : s \Rightarrow t}$$

[Appl]

$$\frac{H \vdash M : s \Rightarrow t \quad H \vdash N : s}{H \vdash MN : t}$$

[Pairing]

$$\frac{H \vdash M : s \quad H \vdash N : t}{H \vdash (M, N) : s \times t}$$

[First]

$$\frac{H \vdash M : s \times t}{H \vdash \mathbf{fst}(M) : s}$$

[Second]

$$\frac{H \vdash M : s \times t}{H \vdash \mathbf{snd}(M) : t}$$

Remarks 2.1.15

- 1. In any type assignment, the type of a constant is the type assigned to it in the term part Σ_0 of the signature.
- 2. The projection rule [Proj] asserts that $H \vdash x : H(x)$ if $x \in H$.
- 3. The application rule [Appl] says that, for an type assignment H, a term of the form MN has type t if there exists a type s such that the operand N has type s and the operator M has type $s \Rightarrow t$.
- 4. The abstraction rule [Abs] is the subtler of the typing rules. Intuitively, the rule says that if M specifies a result of type t for each variable x:s, then the expression $\lambda x:s.M$ defines a function of type $s\Rightarrow t$ (other free variables of M are not affected by the λ -abstraction and must have their types in H). Note, however, the relation between the conclusion of the rule, where x represents a binding variable in the right side of \vdash , and its hypothesis, where x is a free variable in M. It is essential that the binding variable in the term representative of the conclusion not appear in H, since otherwise, H, x:s would not be a well-formed type assignment, since x is not free in $\lambda x:s.M$.
- 5. The rule [Pairing] says that the type of a pair (M, N) is $s \times t$ only if s is the type of M and t is the type of N.
- 6. The rules [First] and [Second] say, respectively, that if a term M has type $s \times t$, then its first coordinate $\mathbf{fst}(M)$ has type s and its second coordinate $\mathbf{snd}(M)$ has type t.

An important part of the calculus is the proof system for deriving equations. The equational proof system also gives origin to a set of reduction rules, which constitutes a simple model of an interpreter of a programming language.

Definition 2.1.16 An equation-in-contex (or equation, for short) in $\lambda^{\to,\times}$ is a 4-tuple (H, M, N, t), where H is a type assignment, M and N are λ -terms, and t is a type

Remark 2.1.17 In order to make a tuple like this more readable, it is helpful to replace the comas separating the components of the tuple for more suggestive symbols, and we will write $(H \triangleright M = N : t)$. The triangular marker is intended to indicate where the interesting part of the tuple begins. The "heart of the tuple" is the pair of terms on either side of the equation symbol; H and t provide typing information about these terms.

Definition 2.1.18 An equational $\lambda^{\to,\times}$ -theory T is a pair (Σ, Ax) , where Σ is a $\lambda^{\to,\times}$ -signature and Ax is a set of equations-in-contex.

Definition 2.1.19 For the judgement that an equation is provable, we define the relation \vdash between theories T and equations $(H \rhd M = N : t)$ as being the least relation satisfying the following rules:

{Axiom}

$$\frac{(H \rhd M = N : t) \in T}{T \vdash (H \rhd M = N : t)}$$

{Add}

$$\frac{T \vdash (H \rhd M = N : t) \quad x \not\in H}{T \vdash (H, x : s \rhd M = N : t)}$$

{Drop}

$$\frac{T \vdash (H, x : s \rhd M = N : t) \quad x \not\in FV(M) \cup FV(N)}{T \vdash (H \rhd M = N : t)}$$

{Permute}

$$\frac{T \vdash (H,x:r,y:s,H' \rhd M = N:t)}{T \vdash (H,y:s,x:r,H' \rhd M = N:t)}$$

{Refl}

$$\frac{H \vdash M : t}{T \vdash (H \rhd M = M : t)}$$

{Sym}

$$\frac{T \vdash (H \rhd M = N : t)}{T \vdash (H \rhd N = M : t)}$$

{Trans}

$$\frac{T \vdash (H \rhd L = M:t) \quad T \vdash (H \rhd M = N:t)}{T \vdash (H \rhd L = N:t)}$$

{Cong}

$$\frac{T \vdash (H \rhd M = M': s \Rightarrow t) \quad T \vdash (H \rhd N = N': s)}{T \vdash (H \rhd MN = M'N': t)}$$

 $\{\xi\}$

$$\frac{T \vdash (H, x : s \rhd M = N : t)}{T \vdash (H \rhd \lambda x : s.M = \lambda x : s.N : s \Rightarrow t)}$$

 $\{\beta\}$

$$\frac{H,x:s\vdash M:t\quad H\vdash N:s}{T\vdash (H\rhd (\lambda x:s.M)(N)=[N/x]M:t)}$$

 $\{\eta\}$

$$\frac{H \vdash M : s {\Rightarrow} t \quad x \not\in FV(M)}{T \vdash (H \rhd \lambda x : s.M(x) = M : s {\Rightarrow} t)}$$

{First}

$$\frac{H \vdash M : s \quad H \vdash N : t}{T \vdash (H \rhd \mathbf{fst}(M, N) = M : s)}$$

{Second}

$$\frac{H \vdash M : s \quad H \vdash N : t}{T \vdash (H \rhd \mathbf{snd}(M, N) = N : t)}$$

{Pairing}

$$\frac{H \vdash M : s \times t}{T \vdash (H \rhd (\mathbf{fst}(M), \mathbf{snd}(M)) = M : s \times t)}$$

Remarks 2.1.20

- 1. The axiom rule $\{Axiom\}$ asserts that all the equations in $T = (\Sigma, Ax)$ are provable from T.
- 2. The addition rule $\{Add\}$ asserts that the typing assignment H in an equation can be extended by assignment of a type to a new variable.
- 3. The drop rule {Drop} asserts that the assignment of a type to a variable can be removed from the type assignment component of an equation if the variable has no free occurrence in either of the term components of the equation.

- 4. By the permutation rule {Permute}, any equation E obtained from an equation E' by permuting the order of variable/type pairs in the assignment component of E' is provable if E' is.
- 5. The reflexivity rule {Refl}, symmetric rule {Sym}, and transitivity rule {Trans} assert that = can be viewed as an equivalence relation on terms, relative to a fixed theory and type tags.
- 6. The congruence rule $\{Cong\}$ and the ξ -rule assert that application and abstraction respectively are congruences relative to the equality relation (relative to a fixed theory and type tags).
- 7. The rules $-\beta$ and $-\eta$ are the key to the λ -calculus. The β -rule asserts that the application of a abstraction to an operand is equal to the result of substituting the operand for the bound variable of the abstraction in the body of the abstraction.
- 8. It is not very simple to explain the η -rule. It says that if M is a function, then the function which takes an argument x and applies M to it, is indistinguishable from M. Note that in the η -rule, the variable x can not have a free occurrence in the term M.
- 9. The rules $\{First\}$ and $\{Second\}$ assert that the coordinates of (M, N) are respectively M and N.
- 10. The rule {Pairing} asserts that if a pair is formed by the first and second coordinate of M, then this pair is equal to M.

Definition 2.1.21 An equation $(H \triangleright M = N : t)$ is **provable** in a theory T just in case $T \vdash (H \triangleright M = N : t)$.

Definition 2.1.22 The assertion $T \vdash (H \triangleright M = N : t)$ is called an equational judgement.

Example 2.1.23 We now give an example of a $\lambda^{\to,\times}$ -theory. Consider the $\lambda^{\to,\times}$ -signature Σ which has ground types *nat* and *bool*, and has the following function symbols (term constants):

- tt:bool,
- ff:bool,
- $C: bool, nat, nat \rightarrow nat,$
- $k_n : nat \text{ for each } n \in \mathbb{N},$
- $S: nat \rightarrow nat$,
- $P: nat \rightarrow nat$,
- $Z: nat \rightarrow nat$.

We then set Ax to consist of the following equations-in-context:

- $x : nat, x' : nat \vdash C(tt, x, x') = x$,
- $x : nat, x' : nat \vdash C(ff, x, x') = x'$,
- $\bullet \vdash S(k_n) = k_{n+1},$
- $\bullet \vdash P(k_0) = k_0,$
- $\bullet \vdash P(k_{n+1}) = k_n,$
- $\bullet \vdash Z(k_0) = tt,$
- $\bullet \vdash Z(k_{n+1}) = ff$

One should think of this theory as having ground types the natural numbers and the booleans (where the latter is a two point set which indicates truth and falsity). The function symbols are to be thought of as truth, falsity, a conditional test for two terms, numeral, successor operation on the natural numbers, a predecessor on the natural numbers an a test for zero.

The fact that only well-typed terms can be proved equal is stated in the following

Proposition 2.1.24 If T is a theory and $T \vdash (H \triangleright M = N : t)$, then $H \vdash M : t$ and $H \vdash N : t$.

Definition 2.1.25 In a category with all binary products and a terminal object, the object $\times (A_1, \ldots, A_n)$ is defined as follows:

$$\begin{array}{l} \times () = 1 \\ \times (A) = 1 \times A \\ \times (A_1, \dots, A_n) = (\times (A_1, \dots, A_{n-1})) \times A_n \end{array}$$

We also define

$$fst: (\times (A_1, \dots, A_{n-1})) \times A_n \to (\times (A_1, \dots, A_{n-1}))$$

$$snd: (\times (A_1, \dots, A_{n-1})) \times A_n \to A_n$$

i.e., fst and snd are projections in the sense of A.3.1.

Definition 2.1.26 We define a generalized form of projection $\pi_i^n : \times (A_1, \dots, A_n) \to A_i$ putting $\pi_n^n = snd$, and $\pi_i^n = \pi_i^{n-1} \circ fst$, when i < n.

There exists another class of morphisms on products, consisting of morphisms that permute the coordinates of a product.

Definition 2.1.27 Consider the products

$$A = \times (A_1, \dots, A_k, A_{k+1}, \dots, A_n)$$

$$A' = \times (A_1, \dots, A_{k+1}, A_k, \dots, A_n)$$

where $1 \leq k < n$. By induction on n, we define a map $\tau_k^n[A]: A \to A'$ as follows:

$$\tau_k^n = \left\{ \begin{array}{ll} \langle \langle fst \circ fst, snd \rangle, snd \circ fst \rangle & \text{if } k+1 = n \\ \tau_k^{n-1} \times id_{A_n} & \text{if } k+1 < n \end{array} \right.$$

Remarks 2.1.28

1. The definition of τ_k^n can be (better) understood by seeing the following commutative diagrams (where k+1=n):

$$\times (A_1, \dots, A_{k-1}, A_k, A_{k+1})$$

$$\uparrow sto fst$$

$$\times (A_1, \dots, A_{k-1}, A_k, A_{k+1})$$

$$\uparrow snd$$

$$\times (A_1, \dots, A_{k-1}, A_k, A_{k+1})$$

$$\downarrow snd fst$$

$$\times (A_1, \dots, A_{k-1}, A_k, A_{k+1})$$

$$\uparrow snd fst$$

$$\uparrow snd fst$$

$$\times (A_1, \dots, A_{k-1}, A_{k+1}, A_k)$$

$$\uparrow snd fst$$

$$\downarrow snd fst$$

$$\uparrow snd fst$$

$$\downarrow snd fst$$

where $h = \langle fst \circ fst, snd \rangle$.

2. In the base case, n = 2, the first clause of the above equation provides the definition (since k = 1).

The following lemma captures the basic relationship between projections and permutations.

Lemma 2.1.29 For each i and k such that $1 < k + 1 \le n$ and i is not equal to k or k + 1 we have:

- 1. $\pi_k^n[A] = \pi_{k+1}^n[A'] \circ \tau_k^n[A]$
- 2. $\pi_{k+1}^n[A] = \pi_k^n[A'] \circ \tau_k^n[A]$
- 3. Moreover, if i is not equal to k or k+1 then

$$\pi_i^n[A] = \pi_i^n[A'] \circ \tau_k^n[A]$$

Remark 2.1.30 These three equations are not difficult to understand intuitively. Equation 1 says that, after permutation, the k+1 coordinate is the same as the k coordinate before permutation. Equation 2 says that, after permutation, the k coordinate is the same as the k+1 coordinate before permutation. Equation 3 says that the i coordinate of the tuple before permutation is the i coordinate of the tuple after permutation, if the i coordinate is not one of the ones permuted.

Proof: All three equations are proved by induction on n.

$$\begin{array}{lll} 1. & \pi_k^n[A] = \pi_{k+1}^n[A'] \circ \tau_k^n[A] \\ \text{Base case, } n = 2 \text{ and } k = 1. \\ & \pi_2^2[A'] \circ \tau_1^2[A] = \\ & = \pi_2^2[A'] \circ \langle \langle fst \circ fst, snd \rangle, snd \circ fst \rangle[A] \\ & = snd[A'] \circ \langle \langle fst \circ fst, snd \rangle, snd \circ fst \rangle[A] \\ & = snd \circ fst \\ & = \pi_{n-1}^n = \pi_1^2 = \pi_k^n[A] \end{array} \qquad \begin{array}{ll} \text{(definition } 2.1.27) \\ \text{(definition } 2.1.26) \\ \text{(definition of } snd) \\ \text{(definition of } fst \text{ and } snd) \\ \end{array}$$

To prove the inductive step for the equation 1, suppose that the lemma holds for n-1. The argument requires consideration of two cases. Assume first that k+1=n. Then we have:

$$\begin{array}{ll} \pi_{k+1}^n[A']\circ\tau_k^n[A] = \\ = \pi_{k+1}^n[A']\circ\langle\langle fst\circ fst, snd\rangle, snd\circ fst\rangle[A] & \text{(definition } 2.1.27) \\ = snd[A']\circ\langle\langle fst\circ fst, snd\rangle, snd\circ fst\rangle[A] & \text{(definition } 2.1.26) \\ = snd\circ fst[A] & \text{(definition of } snd) \\ = \pi_{n-1}^n[A] & \text{(definition of } fst \text{ and } snd) \\ = \pi_k^n[A] & \text{(definition of } fst \text{ and } snd) \\ = \pi_k^n[A] & \text{(k+1=n)} \end{array}$$

On the other hand, if k + 1 < n, then

$$\begin{array}{lll} \pi_{k+1}^n[A'] \circ \tau_k^n[A] = \\ = \pi_{k+1}^n[A'] \circ (\tau_k^{n-1} \times id_{A_n})[A] & \text{(definition } 2.1.27) \\ = (\pi_{k+1}^{n-1}[A'] \circ fst) \circ (\tau_k^{n-1} \times id_{A_n})[A] & \text{(definition } 2.1.26) \\ = (\pi_{k+1}^{n-1}[A'] \circ fst) \circ \langle \tau_k^{n-1} \circ fst, id_{A_n} \circ snd \rangle[A] & \text{(definition } A.3.2) \\ = \pi_{k+1}^{n-1}[A'] \circ (fst \circ \langle \tau_k^{n-1} \circ fst, id_{A_n} \circ snd \rangle)[A] & \text{(associativity)} \\ = (\pi_{k+1}^{n-1}[A'] \circ \tau_k^{n-1}) \circ fst[A] & \text{(definition of } fst \text{ and associativity)} \\ = \pi_k^{n-1} \circ fst[A] & \text{(inductive hypothesis)} \\ = \pi_k^n[A] & \text{(definition } 2.1.26) \end{array}$$

2.
$$\pi_{k+1}^n[A] = \pi_k^n[A'] \circ \tau_k^n[A]$$

Base case, n = 2, k = 1. Then:

```
\begin{array}{ll} \pi_1^2[A'] \circ \tau_1^2[A] = \\ = \pi_1^2[A'] \circ \langle \langle fst \circ fst, snd \rangle, snd \circ fst \rangle [A] & \text{(definition } 2.1.27) \\ = \pi_1^1[A'] \circ fst \circ \langle \langle fst \circ fst, snd \rangle, snd \circ fst \rangle [A] & \text{(definition } 2.1.26) \\ = \pi_1^1[A'] \circ \langle fst \circ fst, snd \rangle [A] & \text{(definition of } fst) \\ = snd[A'] \circ \langle fst \circ fst, snd \rangle [A] & \text{(definition } 2.1.26) \\ = snd[A] & \text{(definition of } snd) \\ = \pi_n^n = \pi_{k+1}^n = \pi_2^2[A] & \text{(definition } 2.1.26, k = 1, n = 2) \end{array}
```

To prove the inductive step for equation 2, suppose that the lemma holds for n-1. The argument requires the consideration of two cases. Assume first that k+1=n. Then:

$$\begin{array}{l} \pi_k^n[A'] \circ \tau_k^n[A] = \\ = \pi_k^n[A'] \circ \langle \langle \mathit{fst} \circ \mathit{fst}, \mathit{snd} \rangle, \mathit{snd} \circ \mathit{fst} \rangle [A] \end{array} \tag{definition 2.1.27}$$

```
=\pi_k^{n-1}[A]\circ fst\circ \langle\langle fst\circ fst, snd\rangle, snd\circ fst\rangle[A] \qquad \qquad \text{(definition 2.1.26)} =\pi_k^{n-1}[A']\circ \langle fst\circ fst, snd\rangle[A] \qquad \qquad \text{(definition of } fst) =\pi_k^k[A']\circ \langle fst\circ fst, snd\rangle[A] \qquad \qquad (k+1=n) =snd[A']\circ \langle fst\circ fst, snd\rangle[A] \qquad \qquad \text{(definition 2.1.26)} =snd[A] \qquad \qquad \text{(definition 2.1.26)} =\pi_n^n[A] \qquad \qquad \text{(definition 2.1.26)} =\pi_n^n[A] \qquad \qquad (k+1=n)
```

Now, if k + 1 < n, we have:

```
\begin{array}{ll} \pi_k^n[A'] \circ \tau_k^n[A] = \\ = \pi_k^n[A'] \circ (\tau_k^{n-1} \times id_{A_n})[A] & \text{(definition of } 2.1.27) \\ = (\pi_k^{n-1}[A'] \circ fst) \circ (\tau_k^{n-1} \times id_{A_n})[A] & \text{(definition } 2.1.26) \\ = (\pi_k^{n-1}[A'] \circ fst) \circ \langle \tau_k^{n-1} \circ fst, id_{A_n} \circ snd \rangle [A] & \text{(definition } A.3.2) \\ = \pi_k^{n-1}[A'] \circ (fst \circ \langle \tau_k^{n-1} \circ fst, id_{A_n} \circ snd \rangle)[A] & \text{(associativity)} \\ = (\pi_k^{n-1}[A'] \circ \tau_k^{n-1}) \circ fst[A] & \text{(definition of } fst \text{ and associativity)} \\ = \pi_{k+1}^{n-1}[A'] \circ fst[A] & \text{(inductive hypothesis)} \\ = \pi_{k+1}^{n}[A] & \text{(definition } 2.1.26) \end{array}
```

Equation 3 can be proved similarly (Note that the base case here doesn't apply since i should not be either equal to k or k+1).

2.2 A categorical semantics for $\lambda^{\to,\times}$

The main idea for the categorical semantics of $\lambda^{\to,\times}$ is to interpret types as objects and terms as morphisms in a cartesian closed category ${\bf C}$. Especially, the reader may have already noticed the relation between the equational rules of the calculus and the axioms that define products. In fact, the relation is quite clear: ${\bf fst}: s\times t\to s$ and ${\bf snd}: s\times t\to t$ are easily understood as the projections associated with the categorical product of s and t. Not considering in the moment the problem of "changing" from application to composition, the equational rules concerning products are essentially the same as the ones related to the definition of categorical product. Less immediate, however, is the connection between β (and η) and the rules defining exponents in cartesian closed categories: the chief problem is that the definition of the calculus is based on the process of substitution for which we do not have any immediate equivalent in category theory. However, we are able to find a solution for such question, and before presenting such solution, let us introduce the following

Definition 2.2.1 A **structure** M in a cartesian closed category C for $\lambda^{\to,\times}$ -signature is given by:

• an assignment of a non-empty object $[\![\mathbf{o}]\!]$ for ground type $\mathbf{o} \in \Sigma_1$, where

- a point $\llbracket c \rrbracket : 1 \to \llbracket \Sigma_0(c) \rrbracket$ (in the sense of A.2.4) for each c in the domain of Σ_0 .
- Now, for every term in $\lambda^{\to,\times}$ we have the following:

Let $H = x_1 : t_1, \ldots, x_n : t_n$ be a type assignment. The mapping $[\cdot]$ is defined on assignments by:

Now suppose that $H \vdash M : t$. Let $A = [\![H]\!]$. Then the interpretation of M : t in the assignment H is a morphism

$$\llbracket H \vdash M : t \rrbracket : A \rightarrow \llbracket t \rrbracket,$$

i.e., the intuitive idea is that a term M:t with free variables in A, should be interpreted as a morphism from A to $[\![t]\!]$.

The meaning function $\llbracket \cdot \rrbracket$ on terms is defined inductively as follows:

- 1. Constants: $\llbracket H \vdash c : t \rrbracket = \llbracket c \rrbracket \circ \mathbf{t}_A \text{ where } t = \Sigma_0(c).$
- 2. **Projection:** $\llbracket H \vdash x_i : t_i \rrbracket = \pi_i^n$.
- 3. Abstraction: $\llbracket H \vdash \lambda x : u.N : u \Rightarrow v \rrbracket = curry(\llbracket H, x : u \vdash N : v \rrbracket)$.
- 4. Application: $\llbracket H \vdash LN : s \rrbracket = eval \circ \langle \llbracket H \vdash L : t \Rightarrow s \rrbracket, \llbracket H \vdash N : t \rrbracket \rangle$.
- 5. Pair: $[\![H \vdash (L, N) : s \times t]\!] = \langle [\![H \vdash L : s]\!], [\![H \vdash N : t]\!] \rangle$.
- 6. First: $\llbracket H \vdash \mathbf{fst}(N) : s \rrbracket = fst \circ \llbracket H \vdash N : s \times t \rrbracket$.
- 7. Second: $\llbracket H \vdash \mathbf{snd}(N) : t \rrbracket = snd \circ \llbracket H \vdash N : s \times t \rrbracket$.

Remarks 2.2.2

- 1. The restriction of an assignment of a non-empty object for each ground type is due to the fact that the elimination of variables by the rule [Drop] with empty base types may lead to absurd deductions, turning the calculus unsound. A detailed discussion for these problem in connection with algebraic theories may be found in [8]. An approach usually used in the traditional literature is to avoid the rule [Drop] and then allow arbitrary structures, i.e., also the ones with empty objects.
- 2. A type constant $\mathbf{o} \in \Sigma_1$ denotes any object $[\![\mathbf{o}]\!]$. A type $s \Rightarrow t$ denotes an exponential object $[\![t]\!]$.
- 3. We assume that Σ and \mathbf{C} are fixed for the rest of our discussion.
- 4. The notation \mathbf{t}_A means the unique morphism from A to the terminal object 1.
- 5. If the term M is a constant, then its interpretation is a composite morphism $A \to 1 \to \llbracket c \rrbracket$, where 1 is a terminal object and $1 \to \llbracket c \rrbracket$ a morphism in the sense of A.2.4.
- 6. If the term M is a variable, then its interpretation is a simple projection from the interpretation of its type.

- 7. If the term M is an abstraction, then its interpretation is the morphism $curry(f): A \to \llbracket v \rrbracket^{\llbracket u \rrbracket}$, where $f = \llbracket H, x: u \vdash N: v \rrbracket: A \times \llbracket u \rrbracket \to \llbracket v \rrbracket$. Note that this is well-defined by the bijection $Hom_{\mathbf{C}}(A \times \llbracket u \rrbracket, \llbracket v \rrbracket) \cong Hom_{\mathbf{C}}(A, \llbracket v \rrbracket^{\llbracket u \rrbracket})$.
- 8. If M is a pair, then its interpretation is a product morphism, where the components are the respective interpretations of the components of the pair.
- 9. The equations for **First** and **Second** are straightforward.

Here is an example of how the semantic equations are unwound.

```
 \begin{split} & \llbracket \emptyset \vdash \lambda x : s.\lambda f : s \Rightarrow t.f(x) : s \Rightarrow (s \Rightarrow t) \Rightarrow t \rrbracket \\ &= curry_{\times (1,\llbracket s \rrbracket)} \llbracket x : s \vdash \lambda f : s \Rightarrow t.f(x) : (s \Rightarrow t) \Rightarrow t \rrbracket \\ &= curry_{1} (curry_{\times (1,\llbracket s \rrbracket)} \llbracket H \vdash f(x) : t \rrbracket) & \text{(by 2.2.1.3)} \\ &= curry_{1} (curry_{\times (1,\llbracket s \rrbracket)} (eval_{\llbracket s \rrbracket,\llbracket t \rrbracket} \circ \langle \llbracket H \vdash f : s \Rightarrow t \rrbracket, \llbracket H \vdash x : s \rrbracket \rangle)) \\ &= curry_{1} (curry_{\times (1,\llbracket s \rrbracket)} (eval_{\llbracket s \rrbracket,\llbracket t \rrbracket} \circ \langle \pi_{2}^{2}, \pi_{1}^{2} \rangle)) & \text{(by 2.2.1.4)} \\ &= curry_{1} (curry_{\times (1,\llbracket s \rrbracket)} (eval_{\llbracket s \rrbracket,\llbracket t \rrbracket} \circ \langle \pi_{2}^{2}, \pi_{1}^{2} \rangle)) & \text{(by 2.2.1.2)} \\ &\text{where } H = x : s, f : s \Rightarrow t. \end{split}
```

Definition 2.2.3 Let M be a structure for a $\lambda^{\to,\times}$ -signature in a cartesian closed category C. Given a equation-in-context $H \vdash M = M'$: t we say that M satisfies the equation-in-context if $\llbracket H \vdash M : t \rrbracket$ and $\llbracket H \vdash M' : t \rrbracket$ are equal morphisms in C, and we write this as

$$\mathsf{M} \models H \vdash M = M : t.$$

We also say that M is a **model** of a $\lambda^{\to,\times}$ -theory $T=(\Sigma,Ax)$ if M satisfies all the equations-in-context in Ax, which is denoted by $M\models Ax$.

We must now prove that this interpretation is sound with respect to the equational rules,. Assuming that our model satisfies the equations of a theory T, we must show that it satisfies all of the equalities derivable (the theorems) using the equational axioms. The proof proceeds by induction on the height of the derivation of an equality: we proceed showing that the conclusion of each equational rule holds for the categorical interpretation under the assumption that its hypotheses do.

Lemma 2.2.4 Suppose

$$H = x_1 : t_1, \dots, x_k : t_k, x_{k+1} : t_{k+1}, \dots, x_n : t_n$$

 $H' = x_1 : t_1, \dots, x_{k+1} : t_{k+1}, x_k : t_k, \dots, x_n : t_n$

are type assignments and $H \vdash M : t$. Then

$$[\![H \vdash M : t]\!] = [\![H' \vdash M : t]\!] \circ \tau_k^n[A].$$

Proof: The proof is by structural induction on M.

Case M is a constant:

Case M is a variable x_i . Define

$$A = \times ([[t_1]], \dots, [[t_k]], [[t_{k+1}]], \dots, [[t_n]])$$

$$A' = \times ([[t_1]], \dots, [[t_{k+1}]], [[t_k]], \dots, [[t_n]])$$

There are three cases depending on the value of i. If i=k, then x_i is the k+1 element in the list H'. Therefore, $\llbracket H' \vdash M : t \rrbracket = \pi_{k+1}^n[A']$. Since $\llbracket H \vdash M : t \rrbracket = \pi_k^n[A]$, the desired result follows immediately from equation 1 of lemma 2.1.29. If i=k+1, then x_i is the k element in the list H'. Therefore, $\llbracket H' \vdash M : t \rrbracket = \pi_k^n[A']$. But $\llbracket H \vdash M : t \rrbracket = \pi_{k+1}^n[A]$, and hence the desired result follows from equation 2 of lemma 2.1.29. In the remaining case, that i is equal to neither k nor k+1, we have that $\llbracket H \vdash M : t \rrbracket = \pi_i^n[A]$ and the equation 3 from lemma 2.1.29 presents the desired conclusion.

Case $M \equiv LN$. Then

$$\begin{split} & \llbracket H \vdash LN : t \rrbracket = \\ & = eval_{\llbracket s \rrbracket, \llbracket t \rrbracket} \circ \langle \llbracket H \vdash L : s \Rightarrow t \rrbracket, \llbracket H \vdash N : s \rrbracket \rangle \\ & = eval_{\llbracket s \rrbracket, \llbracket t \rrbracket} \circ \langle \llbracket H' \vdash L : s \Rightarrow t \rrbracket \circ \tau_k^n, \llbracket H' \vdash N : s \rrbracket \circ \tau_k^n \rangle \\ & = eval_{\llbracket s \rrbracket, \llbracket t \rrbracket} \circ \langle \llbracket H' \vdash L : s \Rightarrow t \rrbracket, \llbracket H' \vdash N : s \rrbracket \rangle \circ \tau_k^n \\ & = \llbracket H' \vdash LN : t \rrbracket \circ \tau_k^n \end{split}$$
 (inductive hypothesis)

Case $M \equiv \lambda y : s.M'$. Then $t \equiv s \Rightarrow v$ for some v. Now,

$$\begin{aligned} \llbracket H \vdash \lambda y : s.M' : s \Rightarrow v \rrbracket &= \\ &= curry_A(\llbracket H, y : s \vdash M' : v \rrbracket) & \text{(by 2.2.1.3)} \\ &= curry_A(\llbracket H', y : s \vdash M' : v \rrbracket \circ \tau_k^n) & \text{(inductive hypothesis)} \\ &= \llbracket H' \vdash \lambda y : s.M' : s \Rightarrow v \rrbracket \circ \tau_k^n & \text{(by 2.2.1.3)} \end{aligned}$$

Case $M \equiv (L, N)$. Then $t \equiv s \times u$ for some s and some u. Now we have

$$\begin{split} \llbracket H \vdash (L,N) : s \times u \rrbracket &= \\ &= \langle \llbracket H \vdash L : s \rrbracket, \llbracket H \vdash N : u \rrbracket \rangle & \text{(by 2.2.1.5)} \\ &= \langle \llbracket H' \vdash L : s \rrbracket \circ \tau_k^n, \llbracket H' \vdash N : u \rrbracket \circ \tau_k^n \rangle & \text{(inductive hypothesis)} \\ &= \langle \llbracket H' \vdash L : s \rrbracket \circ, \llbracket H' \vdash N : u \rrbracket \rangle \circ \tau_k^n & \text{(by A.3.6)} \\ &= \llbracket H' \vdash (L,N) : s \times u \rrbracket \circ \tau_k^n & \text{(definition 2.2.1.5)} \end{split}$$

Case $M \equiv \mathbf{fst}(N)$. Then $N: t \times u$ for some u. Now we have

The second projection rule has a similar proof.

Proposition 2.2.5 The categorical model is sound with respect to the rule {Permute}.

Proof: Suppose $H = H_0, x : r, y : s, H_1$ and $H' = H_0, y : s, x : r, H_1$ are type assignments, M, N terms such that $H \vdash M : t$ and $H \vdash N : t$ and that $\llbracket H \vdash M : t \rrbracket = \llbracket H \vdash N : t \rrbracket$. Now, if y is the k'th element of H and the length of H is n, we have that

$$\begin{split} & \llbracket H' \vdash M : t \rrbracket = \\ & = \llbracket H \vdash M : t \rrbracket \circ \tau_k^n \llbracket H' \rrbracket \\ & = \llbracket H \vdash N : t \rrbracket \circ \tau_k^n \llbracket H' \rrbracket \\ & = \llbracket H' \vdash N : t \rrbracket \\ & \text{(assumption)} \\ & = \llbracket H' \vdash N : t \rrbracket \\ \end{aligned}$$

Lemma 2.2.6 If x is not free in M, then $\llbracket H, x : s \vdash M : t \rrbracket = \llbracket H \vdash M : t \rrbracket \circ fst$.

Proof: The proof proceeds by structural induction on M. Suppose $[\![s]\!] = B, [\![t]\!] = C$ and $[\![H,x:s \vdash M:t]\!] : A \times B \to C$.

Case $M \equiv c$. Then we have

Case $M \equiv x_i \not\equiv x$ where $H = x_1 : t_1, \dots, x_n : t_n$. Then we have:

Case $M \equiv LN$:

Case $M \equiv \lambda y : u.M'$. Then $t \equiv u \Rightarrow v$ for some v. Let $\tau = \langle \langle fst \circ fst, snd \rangle, snd \circ fst \rangle : (A \times B) \times U \rightarrow (A \times U) \times B$, where $U = \llbracket u \rrbracket$. Now, see that

```
\llbracket H, x : s \vdash \lambda y : u.M' : u \Rightarrow v \rrbracket =
= curry_{\times (A, \llbracket s \rrbracket)}(\llbracket H, x:s, y:u \vdash M':v \rrbracket)
                                                                                                                                                                              (by 2.2.1.3)
= curry_{\times (A, \llbracket s \rrbracket)}(\llbracket H, y : u, x : s \vdash M' : v \rrbracket \circ \tau)
                                                                                                                                                                                 (by 2.2.4)
= curry_{\times (A, \llbracket s \rrbracket)}(\llbracket H, y : u \vdash M' : v \rrbracket \circ fst \circ \tau)
                                                                                                                                                      (inductive hypothesis)
= curry_{\times (A, \llbracket s \rrbracket)}(\llbracket H, y : u \vdash M' : v \rrbracket \circ fst \circ \langle \langle fst \circ fst, snd \rangle, snd \circ fst \rangle)
                                                                                                                                                                         (substitution)
= curry_{\times (A, \llbracket s \rrbracket)}(\llbracket H, y : u \vdash M' : v \rrbracket \circ \langle \mathit{fst} \circ \mathit{fst}, \mathit{snd} \rangle)
                                                                                                                                                                 (definition of fst)
= curry_{\times (A, \llbracket s \rrbracket)}(\llbracket H, y : u \vdash M' : v \rrbracket \circ \langle fst \circ fst, id_U \circ snd \rangle)
                                                                                                                   (identity; fst \circ fst : (A \times B) \times U \to A)
= curry_{\times (A, \llbracket s \rrbracket)}(\llbracket H, y : u \vdash M' : v \rrbracket \circ (fst \times id_U))
                                                                                                                                          (by A.3.2; fst: A \times B \rightarrow A)
= curry_A(\llbracket H, y : u \vdash M' : v \rrbracket) \circ fst
                                                                                                                                                                              (by 1.2.10)
= [H \vdash \lambda y : u.M' : u \Rightarrow v] \circ fst
                                                                                                                                                                              (by 2.2.1.3)
```

Case $M \equiv (L, N)$. Then $t \equiv s \times u$ for some s and for some v. Now we have

$$\begin{split} \llbracket H,x:s \vdash (L,N):s \times u \rrbracket &= \\ &= \langle \llbracket H,x:s \vdash L:s \rrbracket, \llbracket H,x:s \vdash N:u \rrbracket \rangle & \text{(by 2.2.1.5)} \\ &= \langle \llbracket H,x:s \vdash L:s \rrbracket \circ fst, \llbracket H,x:s \vdash N:u \rrbracket \circ fst \rangle & \text{(inductive hypothesis)} \\ &= \langle \llbracket H \vdash L:s \rrbracket, \llbracket H \vdash N:u \rrbracket \rangle \circ fst & \text{(by A.3.6)} \\ &= \llbracket H \vdash (L,N):s \times u \rrbracket \circ fst & \text{(by 2.2.1.5)} \end{split}$$

Case $M \equiv \mathbf{fst}(N)$. Then $N: t \times u$ for some u. Now we have:

The second projection rule has a similar proof.

Proposition 2.2.7 The categorical model is sound with respect to the rule $\{\eta\}$.

```
Proof: Suppose H \vdash \lambda x : s.M(x) : s \Rightarrow t, x \notin FV(M) and that  \llbracket H \vdash \lambda x : s.M(x) : s \Rightarrow t \rrbracket : A \to C^B
```

where $B = \llbracket s \rrbracket$ and $C = \llbracket t \rrbracket$. Then we have:

```
\llbracket H \vdash \lambda x : s.M(x) : s \Rightarrow t \rrbracket =
= curry_A(H, x : s \vdash M(x) : t]
                                                                                                                                                                                                                       (by 2.2.1.3)
= curry_A(eval_{\llbracket s \rrbracket, \llbracket t \rrbracket} \circ \langle \llbracket H, x : s \vdash M : s \Rightarrow t \rrbracket, \llbracket H, x : s \vdash x : s \rrbracket \rangle)
                                                                                                                                                                                                                       (by 2.2.1.4)
= curry_A(eval_{\llbracket s \rrbracket, \llbracket t \rrbracket} \circ \langle \llbracket H, x : s \vdash M : s \Rightarrow t \rrbracket, \pi_{n+1}^{n+1} \rangle)
                                                                                                                                                                                                                       (by 2.2.1.2)
= curry_A(eval_{\llbracket s \rrbracket, \llbracket t \rrbracket} \circ \langle \llbracket H, x : s \vdash M : s \Rightarrow t \rrbracket, snd \rangle)
                                                                                                                                                                                                                        (by 2.1.26)
= curry_A(eval_{\llbracket s \rrbracket, \llbracket t \rrbracket} \circ \langle \llbracket H \vdash M : s \Rightarrow t \rrbracket \circ \mathit{fst}, \mathit{snd} \rangle)
                                                                                                                                                                                                                            (by 2.2.6)
= curry_A(eval_{\llbracket s \rrbracket, \llbracket t \rrbracket} \circ \langle \llbracket H \vdash M : s \Rightarrow t \rrbracket \circ \mathit{fst}, id_B \circ \mathit{snd} \rangle)
                                                                                                                                                                                                                            (identity)
= curry_A(eval_{\llbracket s \rrbracket, \llbracket t \rrbracket} \circ (\llbracket H \vdash M : s \Rightarrow t \rrbracket \times id_B))
                                                                                                                                                                                                                           (by A.3.2)
= \llbracket H \vdash M : s \Rightarrow t \rrbracket
                                                                                                                                                                                                                            (by 1.2.7)
```

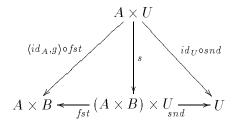
Lemma 2.2.8 *Let*

$$\tau = \langle \langle fst \circ fst, snd \rangle, snd \circ fst \rangle : (A \times U) \times B \to (A \times B) \times U$$

Then, for any morphism $g: A \to B$,

$$\tau \circ \langle id_{A \times U}, g \circ fst \rangle = \langle id_A, g \rangle \times id_U$$

Proof: Consider the following diagram:



Putting $s = \langle id_A, g \rangle \times id_U$, we have that

$$fst \circ \langle id_A, g \rangle \times id_U \rangle =$$

$$= fst \circ \langle \langle id_A, g \rangle \circ fst, id_U \circ snd \rangle$$

$$= \langle id_A, g \rangle \circ fst$$
 (definition of fst)

Similarly, $snd \circ \langle \langle id_A, g \rangle \circ fst, id_U \circ snd \rangle = id_U \circ snd$ and thus the above diagram commutes. On the other hand, $s = \tau \circ \langle id_{A \times U}, g \circ fst \rangle$ also does the job since

```
\begin{array}{ll} fst \circ \tau \circ \langle id_{A \times U}, g \circ fst \rangle \\ = fst \circ \tau \circ \langle id_{A} \times id_{U}, g \circ fst \rangle \\ = fst \circ \tau \circ \langle \langle id_{A} \circ fst, id_{U} \circ snd \rangle, g \circ fst \rangle \\ = fst \circ \tau \circ \langle \langle id_{A} \circ fst, id_{U} \circ snd \rangle, g \circ fst \rangle \\ = fst \circ \langle \langle fst \circ fst, snd \rangle, snd \circ fst \rangle \circ \langle \langle id_{A} \circ fst, id_{U} \circ snd \rangle, g \circ fst \rangle \\ = \langle fst \circ fst, snd \rangle \circ \langle \langle id_{A} \circ fst, id_{U} \circ snd \rangle, g \circ fst \rangle \\ = \langle id_{A} \circ fst, g \circ fst \rangle \\ = \langle id_{A}, g \rangle \circ fst \end{array} \qquad \begin{array}{l} \text{(by A.3.6)} \\ \text{(definition of } fst \text{ and } snd) \\ \text{(by A.3.6)} \end{array}
```

Similarly, we have that

$$\begin{array}{ll} snd \circ \tau \circ \langle id_{A \times U}, g \circ fst \rangle = \\ = snd \circ \tau \langle id_A \times id_U, g \circ fst \rangle & \text{(by A.3.7)} \\ = snd \circ \tau \langle \langle id_A \circ fst, id_U \circ snd \rangle, g \circ fst \rangle & \text{(by A.3.2)} \\ = snd \circ \langle \langle fst \circ fst, snd \rangle, snd \circ fst \rangle \circ \langle \langle id_A \circ fst, id_U \circ snd \rangle, g \circ fst \rangle & \text{(substitution)} \\ = snd \circ fst \circ \langle \langle id_A \circ fst, id_U \circ snd \rangle, g \circ fst \rangle & \text{(definition } snd) \\ = id_U \circ snd & \text{(definition } fst \text{ and } snd) \end{array}$$

However, by the universal property of $(A \times B) \times U$, s must be unique, and thus

```
\tau \circ \langle id_{A \times U}, g \circ fst \rangle = s = \langle id_A, g \rangle \times id_U
```

Lemma 2.2.9 (Substitution Lemma). Let

$$f = \llbracket H, x : s \vdash M : t \rrbracket : A \times B \to C \ e$$
$$g = \llbracket H \vdash N : s \rrbracket : A \to B.$$

where $B = \llbracket s \rrbracket$ and $C = \llbracket t \rrbracket$. Then $\llbracket H \vdash [N/x]M : t \rrbracket = f \circ \langle id_A, g \rangle : A \to C$.

Proof: The proof proceeds by structural induction on M.

Case $M \equiv c$:

Case $M \equiv x$. Then $s \equiv t$, B = C, and we have

$$\begin{split} \llbracket H \vdash [N/x]x : t \rrbracket &= \\ &= \llbracket H \vdash N : s \rrbracket & \text{(by 2.1.7)} \\ &= g & \text{(definition of } g) \\ &= snd \circ \langle id_A, g \rangle & \text{(definition of } snd) \\ &= \pi_{n+1}^{n+1} \circ \langle id_A, g \rangle & \text{(by 2.1.26, } \langle id_A, g \rangle : A \to A \times B, \pi_{n+1}^{n+1} : A \times B \to B) \\ &= \llbracket H, x : s \vdash x : t \rrbracket \circ \langle id_A, g \rangle & \text{(by 2.2.1.2)} \\ &= f \circ \langle id_A, g \rangle & \text{(definition of } f) \end{split}$$

Case $M \equiv x_i$, where $H = x_1 : t_1, \dots, x_n : t_n$, and $x_i \not\equiv x$. Then we have:

$$\begin{split} \llbracket H \vdash \llbracket N/x \rrbracket x_i : t_i \rrbracket &= \\ &= \llbracket H \vdash x_i : t_i \rrbracket = \\ &= \pi_i^n & \text{(by 2.1.7)} \\ &= \pi_i^{n+1} \circ \langle id_A, g \rangle & (\pi_i^n : A \to C, \pi_i^{n+1} : A \times B \to C) \\ &= \llbracket H, x : s \vdash x_i : t_i \rrbracket \circ \langle id_A, g \rangle & \text{(by 2.2.1.2)} \\ &= f \circ \langle id_A, g \rangle & \text{(definition of } f) \end{split}$$

Case $M \equiv M_0 M_1$:

$$\begin{split} \llbracket H \vdash \llbracket N/x \rrbracket M_0 M_1 : t \rrbracket &= \\ &= \llbracket H \vdash \llbracket N/x \rrbracket M_0 \llbracket N/x \rrbracket M_1 \rrbracket \\ &= eval_{\llbracket u \rrbracket, \llbracket t \rrbracket} \circ \langle \llbracket H \vdash \llbracket N/x \rrbracket M_0 : u \Rightarrow t \rrbracket, \llbracket H \vdash \llbracket N/x \rrbracket M_1 : u \rrbracket \rangle \\ &= eval_{\llbracket u \rrbracket, \llbracket t \rrbracket} \circ \langle \llbracket H, x : s \vdash M_0 : u \Rightarrow t \rrbracket \circ \langle id_A, g \rangle, \llbracket H, x : s \vdash M_1 : u \rrbracket \circ \langle id_A, g \rangle \rangle \\ &= (eval_{\llbracket u \rrbracket, \llbracket t \rrbracket} \circ \langle \llbracket H, x : s \vdash M_0 : u \Rightarrow t \rrbracket, \llbracket H, x : s \vdash M_1 : u \rrbracket \rangle) \circ \langle id_A, g \rangle \\ &= \llbracket H, x : s \vdash M_0 M_1 : t \rrbracket \circ \langle id_A, g \rangle \end{aligned} \qquad \text{(by A.3.6)} \\ &= f \circ \langle id_A, g \rangle \qquad \text{(definition of } f)$$

$$A \xrightarrow{\langle id_A, g \rangle} A \times \llbracket s \rrbracket \xrightarrow{f} \llbracket t \rrbracket$$

Figure 1: Figure to illustrate the process of substitution.

Case $M \equiv \lambda y : u.M'$, where $t \equiv u \Rightarrow v$ for some v. Let

$$\tau = \langle \langle \mathit{fst} \circ \mathit{fst}, \mathit{snd} \rangle, \mathit{snd} \circ \mathit{fst} \rangle : (A \times U) \times B \to (A \times B) \times U$$

Now,

```
\llbracket H \vdash \lceil N/x \rceil \lambda y : u.M' : u \Rightarrow v \rrbracket =
= \llbracket H \vdash \lambda y : u . \lceil N/x \rceil M' : u \Rightarrow v \rrbracket
                                                                                                                                                          (definition 2.1.7)
= curry_A(\llbracket H, y : u \vdash \lceil N/x \rceil M' : v \rrbracket)
                                                                                                                                                                     (by 2.2.1.3)
= curry_A(\llbracket H, y : u, x : s \vdash M' : v \rrbracket \circ \langle id_{A \times U}, \llbracket H, y : u \vdash N : s \rrbracket \rangle)
                                                                                                                                               (inductive hypothesis)
= curry_A(\llbracket H, y : u, x : s \vdash M' : v \rrbracket \circ \langle id_{A \times U}, \llbracket H \vdash N : s \rrbracket \circ fst \rangle)
                                                                                                     (Remark 2.1.10.2 \Rightarrow y \notin FV(N), by 2.2.6)
= curry_A(\llbracket H, y : u, x : s \vdash M' : v \rrbracket \circ \langle id_{A \times U}, g \circ fst \rangle)
                                                                                                                                                            (definition of g)
= curry_A(\llbracket H, x:s,y:u \vdash M':v \rrbracket \circ \tau \circ \langle id_{A\times U}, g \circ fst \rangle)
                                                                                                                                                                         (by 2.2.4)
= curry_A(\llbracket H, x : s, y : u \vdash M' : v \rrbracket \circ (\langle id_A, g \rangle \times id_U))
                                                                                                                                                                         (by 2.2.8)
= curry_{\times (A, \llbracket s \rrbracket)}(\llbracket H, x:s,y:u \vdash M':v \rrbracket) \circ \langle id_A, g \rangle
                                                                                                                                                                      (by 1.2.10)
= \llbracket H, x : s \vdash \lambda y : u.M' : u \Rightarrow v \rrbracket \circ \langle id_A, g \rangle
                                                                                                                                                                     (by 2.2.1.3)
= f \circ \langle id_A, q \rangle
                                                                                                                                                            (definition of f)
```

Case $M \equiv (M_0, M_1)$. Then $t = s \times u$ for some s and some u. Now we have:

```
 \begin{split} & \llbracket H \vdash [N/x](M_0, M_1) : s \times u \rrbracket = \\ & = H \vdash ([N/x]M_0, [N/x]M_1) : s \times u \rrbracket \\ & = \langle \llbracket H \vdash [N/x]M_0 : s \rrbracket, \llbracket H \vdash [N/x]M_1 : u \rrbracket \rangle \\ & = \langle \llbracket H, x : s \vdash M_0 : s \rrbracket \circ \langle id_A, g \rangle, \llbracket H, x : s \vdash M_1 : u \rrbracket \circ \langle id_A, g \rangle \rangle \\ & = \langle \llbracket H, x : s \vdash M_0 : s \rrbracket \rangle, \llbracket H, x : s \vdash M_1 : u \rrbracket \rangle \circ \langle id_A, g \rangle \end{aligned}  (inductive hypothesis)  = \langle \llbracket H, x : s \vdash M_0 : s \rrbracket \rangle, \llbracket H, x : s \vdash M_1 : u \rrbracket \rangle \circ \langle id_A, g \rangle  (by A.3.6)  = \llbracket H, x : s \vdash (M_0, M_1) : s \times u \rrbracket \circ \langle id_A, g \rangle  (by 2.2.1.5)
```

The proofs for the projection rules are trivial.

Remark 2.2.10 The effect of the substitution of N:s by x:s in M:t is obtained simply by the composition of $\langle id_A, \llbracket H \vdash N:s \rrbracket \rangle = \langle id_A, g \rangle$ with $\llbracket H, x:s \vdash M:t \rrbracket = f$ (see Figure 1).

Proposition 2.2.11 The categorical model is sound with respect to the rule $\{\beta\}$.

Proof: Suppose $H \vdash \lambda x : s.M : s \Rightarrow t$ and $H \vdash N : s$. Then

Proposition 2.2.12 The categorical model is sound with respect to the rules {First} and {Second}.

Proof: To prove soundness of the first projection rule ({First}), suppose $H \vdash M : s$ and $H \vdash N : t$. Then

The soundness of the rule {Second} can be shown in a similar way, i.e.,

Proposition 2.2.13 The categorical model is sound with respect to the rule {Pairing}.

Proof: Suppose $H \vdash M : s \times t$. Then,

$$\begin{split} & \llbracket H \vdash (\mathbf{fst}(M), \mathbf{snd}(M)) : s \times t \rrbracket = \\ & = \langle \llbracket H \vdash \mathbf{fst}(M) : s \rrbracket, \llbracket H \vdash \mathbf{snd}(M) : t \rrbracket \rangle \\ & = \langle fst \circ \llbracket H \vdash M : s \times t \rrbracket, snd \circ \llbracket H \vdash M : s \times t \rrbracket \rangle \\ & = \llbracket H \vdash M : s \times t \rrbracket \end{aligned}$$
 (by 2.2.1.5) (by 2.2.1.6 and 2.2.1.7) (by A.3.4)

Proposition 2.2.14 The categorical model is sound with respect to the rule $\{\xi\}$.

Proof: Suppose that $\llbracket H, x : s \vdash M : t \rrbracket$ and $\llbracket H, x : s \vdash N : t \rrbracket$. Then $\llbracket H \vdash \lambda x : s.M : s \Rightarrow t \rrbracket =$

$$= curry(\llbracket H, x : s \vdash N : t \rrbracket)$$

$$= \llbracket H \vdash \lambda x : s.N : s \Rightarrow t \rrbracket$$
(assumption)
(by 2.2.1.3)

Lemma 2.2.15 For each type t, the object [t] is non-empty.

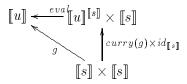
Proof: According to A.2.4, we must show that for each type t, there is a morphism $f: 1 \to [\![t]\!]$ in \mathbb{C} , where \mathbb{C} is a cartesian closed category. The proof proceeds by structural induction on t.

Case $t \equiv \Sigma_0(c)$, where c is a constant. Then the lemma holds, since by definition 2.2.1, we have a point $[\![c]\!]: 1 \to [\![\Sigma_0(c)]\!]$ for each c in the domain of Σ_0 .

Case $t \equiv \mathbf{o} \in \Sigma_1$, i.e., t is a type constant. According to definition 2.2.1, $[\mathbf{o}]$ is a non-empty object, and hence there must be a morphism, say f, from 1 to $[\mathbf{o}]$.

Case $t \equiv s \times u$ for some type s and some type u. By inductive hypothesis, we have morphisms $f: 1 \to \llbracket s \rrbracket$ and $g: 1 \to \llbracket u \rrbracket$. Now, since $\mathbf C$ is a cartesian closed category, we have a product $\llbracket s \rrbracket \times \llbracket u \rrbracket$ with the projections $\pi_{\llbracket s \rrbracket} : \llbracket s \rrbracket \times \llbracket u \rrbracket \to \llbracket s \rrbracket$ and $\pi_{\llbracket u \rrbracket} : \llbracket s \rrbracket \times \llbracket u \rrbracket \to \llbracket u \rrbracket$. Then, by the universal property of the product $\llbracket s \rrbracket \times \llbracket u \rrbracket$, there must be a unique morphism $\langle f, g \rangle : 1 \to \llbracket s \rrbracket \times \llbracket u \rrbracket$, such that $\pi_{\llbracket s \rrbracket} \circ \langle f, g \rangle = f$ and $\pi_{\llbracket u \rrbracket} \circ \langle f, g \rangle = g$.

Case $t \equiv s \Rightarrow u$ for some type s and for some type u. Since \mathbf{C} is a cartesian closed category, we have that for any $g: \llbracket s \rrbracket \times \llbracket s \rrbracket \to \llbracket u \rrbracket$ there exists a unique $curry(g): \llbracket s \rrbracket \to \llbracket u \rrbracket^{\llbracket s \rrbracket}$ (note that the product $\llbracket s \rrbracket \times \llbracket s \rrbracket$ is well-defined since \mathbf{C} is cartesian closed) such that the following diagram commutes:



Now, by inductive hypothesis, there exists a morphism $f: 1 \to \llbracket s \rrbracket$ and hence that $curry(g) \circ f$ is a morphism from 1 to $\llbracket u \rrbracket^{\llbracket s \rrbracket}$.

Proposition 2.2.16 The categorical model is sound with respect to to the rule {Drop}.

Proof: Suppose that $\llbracket H, x : s \vdash M : t \rrbracket : A \times B \to C$ and that $x \notin FV(M) \cup FV(N)$. Since $B = \llbracket s \rrbracket$ is non-empty (see 2.2.15), there exists a morphism $q : 1 \to \llbracket s \rrbracket$ (see A.2.4). Now, observe that

$$\begin{split} \llbracket H \vdash M : t \rrbracket &= \\ &= \llbracket H \vdash M : t \rrbracket \circ \mathit{fst} \circ \langle \mathit{id}_A, q \circ \mathbf{t}_A \rangle \\ &\qquad \qquad (q \circ \mathbf{t}_A : A \to 1 \to \llbracket s \rrbracket, \langle \mathit{id}_A, q \circ \mathbf{t}_A \rangle : A \to A \times \llbracket s \rrbracket, \, \mathsf{definition of } \mathit{fst}) \end{split}$$

```
 = \llbracket H, x : s \vdash M : t \rrbracket \circ \langle id_A, q \circ \mathbf{t}_A \rangle  (by 2.2.6)  = \llbracket H, x : s \vdash N : t \rrbracket \circ \langle id_A, q \circ \mathbf{t}_A \rangle  (assumption)  = \llbracket H \vdash N : t \rrbracket \circ fst \circ \langle id_A, q \circ \mathbf{t}_A \rangle  (by 2.2.6)  = \llbracket H \vdash N : t \rrbracket \circ id_A  (definition of fst)  = \llbracket H \vdash N : t \rrbracket  (identity)
```

The proofs for {Refl}, {Sym}, {Trans} and {Cong} are immediate. It remains only the {Add} rule.

Lemma 2.2.17 If $H \vdash M : t$ and x does not appear in H, then $x \notin FV(M)$.

Proof: If M is a variable y, then it can not be x, since $H \vdash M : t$ must follow from [Proj]. Hence $y \in H$. Suppose now that the lemma holds for subterms of M, i.e., suppose that for any subterm M' of M, if $H' \vdash M' : t$ and x' is a variable which does not appear in H', then $x' \notin FV(M')$. Now the goal is to prove that the lemma holds for M itself.

If M is an abstraction, then M must have the form $\lambda y:r.M'$ where $H,y:r\vdash M':s$ and $t\equiv r\Rightarrow s$. If it is the case that this y is the same that x, then the conclusion is immediate, since $x\not\in FV(\lambda y:r.M')$. On the other hand, if y is different from x, then x does not appear in H,y:r, since by inductive assumption, we know that the preposition holds for for subterms of M, and this means that it holds for M'. Therefore, $x\not\in FV(M')$ and hence, $x\not\in FV(M)$.

If M is an application, then M must have the form LN. Since the application rule [Appl] is the only way to prove that this term has a type, we must have $H \vdash L : s \Rightarrow t$ and $H \vdash N : s$ for some type s. The inductive hypothesis says that the lemma is true for these two terms of M. Thus $x \notin FV(L)$ and $x \notin FV(N)$. Hence, $x \notin FV(LN)$.

If M is a pair, M must have the form (M_0, M_1) . Since the pairing rule [Pairing] is the only way to prove that this term has a type, we must have $H \vdash M_0 : s$ and $H \vdash M_1 : t$ for some s and some t. By inductive hypothesis, $x \notin FV(M_0)$ and $x \notin FV(M_1)$. Hence, $x \notin (M_0, M_1)$. The proofs for the projections are obvious.

Proposition 2.2.18 The categorical model is sound with respect to the rule {Add}.

Proof; Suppose that $H \vdash M : t$, $H \vdash N : t$ and $x \notin H$. Then we have

Theorem 2.2.19 (Soundness for CCC-models). Let C be a cartesian closed category, $T = (\Sigma, Ax)$ a $\lambda^{\to, \times}$ -theory and M a model for T in C. If $T \vdash (H \vdash M = N : t)$ then

 $\mathsf{M} \models H \vdash M = N : t.$

3 Bibliographical remarks

The first section is just a brief exposure to cartesian closed categories. Our presentation is based on [2], [9], and [10]. Especially, examples 1.2.4 and 1.2.5 are borrowed from [9] and [3], respectively. A more complete account on the subject can be found in [13]. Most parts of the material presented on this report are based mainly on [10], chapters 2 and 3, and [2], chapter 8. An extremely succint presentation on the categorical model of the λ -calculus can be found in [17] and [18]. The standard reference for the λ -calculus is [4]. A very good reference for the main type systems used in programming languages is the handbook article [16]. [19] is a pleasing detailed introduction on type theory, and [20], chapter 3, explains in detail that the proof techniques used in this chapter (i.e., structural induction and induction on the height of a derivation tree) are special cases of well-founded induction. The textbook [6] on categorical type theory is simply superb and a must for everyone interested on this topic.

Acknowledgments: This work was developed within a German-Brazil cooperation on Information Technology, in the context of GRAPHIT project, and partially supported by a CNPq-grant 200529/94-3 for Álfio Martini. This is part of a work which was initiated by Prof. Dr. Harmut Ehrig's Kloosterman Lecture "On the role of Category Theory in Computer Science" in Leiden, Holland, September, 1993, as well as the encouragement of Prof. Dr. Daltro Nunes concerning categorical methods in computer science. A very special "thank you" goes to Uwe Wolter, who carefully read this report, found numerous significant errors, and made very useful suggestions on how to correct them.

A Categorical Background

In this appendix we summarize some basic notions from category theory which are used in this report as well as a set of elementary results (propositions and theorems) concerning categorical constructions that are referenced in the proofs. The complete proofs of the following elementary results, may be found, for instance, in [15]. Detailed introductions to this subject may be found in [1], [5], and [14].

A.1 Categories

Definition A.1.1 A category C comprises

- 1. a collection $Ob(\mathbf{C})$ of **objects**;
- 2. a collection $Mor(\mathbf{C})$ of morphisms (also called arrows in the literature);
- 3. two operations dom, cod : $Mor(\mathbf{C}) \to Ob(\mathbf{C})$ assigning to each morphism f two objects, called respectively domain and codomain of f;
- 4. a composition operator $\circ : Mor(\mathbf{C}) \times Mor(\mathbf{C}) \to Mor(\mathbf{C})$ assigning to each pair of morphisms $\langle f, g \rangle$ with dom(g) = cod(f) a composite morphism $g \circ f : dom(f) \to cod(g)$, such that the following associative law holds:

For any morphisms f, g, h in $Mor(\mathbf{C})$ such that $cod(f) = dom(g) \wedge cod(g) = dom(h)$:

$$h \circ (f \circ q) = (h \circ q) \circ h$$

5. an operator $id: Ob(\mathbf{C}) \to Mor(\mathbf{C})$, assigning to each object A, an identity morphism $id_A: A \to A$, such that the following **identity law** holds:

For any morphism f such that $dom(f) = A \wedge cod(f) = B$:

$$id_B \circ f = f \wedge f \circ id_A = f$$

Remarks A.1.2

- 1. Category theory is based on composition as a fundamental operation, in much the same way that classical set theory is based on the "element of" or membership relation.
- 2. Categories will be denoted by uppercase boldface letters $\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$ from the beginning of the alphabet.
- 3. We use letters A, B, \ldots, Y, Z from the alphabet (with subscripts when appropriate) to denote objects and lowercase letters $a, b, c, \ldots, f, g, h, \ldots, y, z$ (occasionally with subscripts) to denote morphisms in any category.
- 4. If $dom(f) = A \wedge cod(f) = B$ we write $f: A \to B$ to denote that f is a morphism from A to B.

CATEGORY	OBJECTS	MORPHISMS
Set	sets	total functions
FinSet	finite sets	total functions
Pfn	sets	partial functions
Rel	sets	binary relations
Mon	monoids	monoid homomorphisms
Poset	posets	monotonic functions
Grp	groups	group homomorphisms
Σ -Alg	Σ -Alg	Σ -homomorphisms
$\mathbf{Cat}(SPEC)$	SPEC-algebras	SPEC-homomorphisms
Aut	finite automata	automata homomorphisms
Graph	directed graphs	graph morphisms

Table 1: Examples of categories

- 5. The collection of all morphisms with domain A and codomain B will be written as $\mathbf{C}(A,B)$ or as $Hom_{\mathbf{C}}(A,B)$.
- 6. Given a category \mathbb{C} , we may write "C-object A" (resp. "C-morphism $f: A \to B$ ") or " $A \in Ob(\mathbb{C})$ " (resp. " $f \in \mathbb{C}(A,B)$ ") to denote that A is a object of \mathbb{C} (resp. f is a morphism from A to B in \mathbb{C}).

Example A.1.3 Table 1 lists some categories by specifying their objects and morphisms.

Example A.1.4 A poset (P, \leq) can be seen as a category $\mathbf{C}(P)$ in the following way: For all $x, y, z \in P$

- 1. An object in $\mathbf{C}(P)$ is a element of P.
- 2. The morphisms of C(P) are defined in the following way:

$$\mathbf{Mor}_{\mathbf{C}(P)}(x,y) = \left\{ \begin{array}{ll} \{x \to y\} & \text{if } x \leq_P y \\ \emptyset & otherwise \end{array} \right.$$

3. The composition of two morphisms $x \to y$ and $y \to z$ is defined by:

$$y \to z \circ x \to y = x \leq_P y \land y \leq_P z.$$

The composition of morphisms in (P, \leq) is associative, which can be seen as follows:

For any $x \to y$, $y \to z$ and $z \to w$ we have:

$$\begin{array}{l} z \to w \circ (y \to z \circ x \to y) = \\ = z \to w \circ (x \leq_P y \wedge y \leq_P z) \\ = (x \leq_P y \wedge y \leq_P z) \wedge z \leq_P w \\ = x \leq_P z \wedge z \leq_P w \end{array} \tag{definition of composition}$$

$$= x \leq_P w \qquad (transitivity \text{ of } \leq_P)$$

$$= x \to w \qquad (definition \text{ of } x \to w)$$
and
$$(z \to w \circ y \to z) \circ x \to y =$$

$$= (y \leq_P z \land z \leq_P w) \circ x \to y \qquad (definition \text{ of composition})$$

$$= x \leq_P y \land (y \leq_P z \land z \leq_P w) \qquad (definition \text{ of composition})$$

$$= x \leq_P y \land y \leq_P w \qquad (transitivity \text{ of } \leq_P)$$

$$= x \leq_P w \qquad (transitivity \text{ of } \leq_P)$$

$$= x \to w \qquad (definition \text{ of } x \to w)$$

4. For each element $x \in P$, there exists a (unique) morphism $x \to x$ (since \leq_P is reflexive), such that for any morphism $x \to y$ the identity axiom is satisfied, i.e.,

$$x \to y \circ x \to x = x \le_P x \land x \le_P y$$
 (definition of composition)
= $x \le_P y$ (transitivity of \le_P)
= $x \to y$ (definition of $x \to y$)

Similarly, it can be shown that $y \to y \circ x \to y = x \to y$.

Example A.1.5 For any pair of categories C and D, we can construct the **product category** $C \times D$ as follows:

- 1. An object in $\mathbf{C} \times \mathbf{D}$ is a pair $\langle A, B \rangle$, where A is an C-object and B is an D-object.
- 2. An $\mathbf{C} \times \mathbf{D}$ -morphism $\langle A_1, B_1 \rangle \to \langle A_2, B_2 \rangle$ is a pair $\langle f, g \rangle$ where $f: A_1 \to A_2$ is an \mathbf{C} -morphism, and $g: B_1 \to B_2$ is an \mathbf{D} -morphism.
- 3. The composition of two morphisms $\langle f_1, g_1 \rangle : \langle A_1, B_1 \rangle \to \langle A_2, B_2 \rangle$ and $\langle f_2, g_2 \rangle : \langle A_2, B_2 \rangle \to \langle A_3, B_3 \rangle$ is a morphism $\langle f_2, g_2 \rangle \circ \langle f_1, g_1 \rangle = \langle f_2 \circ f_1, g_2 \circ g_1 \rangle : \langle A_1, B_1 \rangle \to \langle A_3, B_3 \rangle$, i.e., the composition is defined "componentweise" according to the composition in \mathbf{C} and \mathbf{D} .

Categorical duality is the process "Reverse all morphisms". To each category \mathbf{C} we associate the opposite category \mathbf{C}^{op} by reversing the direction of all morphisms in \mathbf{C} . More formally, we have the following

Definition and proposition A.1.6 For any category C, the dual (or opposite) category of C, is the category C^{op} , where $Ob(C) = Ob(C^{op})$, and for each morphism $f : A \to B$ in C we have a corresponding morphism $f^{op} : B \to A$ in C^{op} such that $cod(f^{op}) = dom(f) \land dom(f^{op}) = cod(f)$. These are the only morphisms in C^{op} . Moreover, composition is defined as follows: for any f^{op} , g^{op} in C^{op} such that $dom(g^{op}) = cod(f^{op})$ one has:

$$q^{op} \circ f^{op} = (f \circ q)^{op}$$

Definition A.1.7 A category C is said to be **small** provided that the collection of all C-objects is a set and not a class. Otherwise it is called **large**.

Example A.1.8 All the categories from Table 1 are large. The category C(P), where P is a poset, is a small category.

Remark A.1.9 If **C** is a small category, then for all $A, B \in Ob(\mathbf{C})$, $Hom_{\mathbf{C}}(A, B)$ is a set, usually called **hom-set** of A and B. We say that a category is **locally small** when for all $A, B \in Ob(\mathbf{C})$, $Hom_{\mathbf{C}}(A, B)$ is a set and not a class. If **C** is a locally small category, it is possible to define some functors from **C** to **Set**, called **hom-functors**, that play a central role in the development of categoy theory.

A.2 Initial and terminal objects

Definition A.2.1 Let C be a category. An object A is a initial object in C if for any C-object A there exists a unique morphism $f \in C(A, B)$

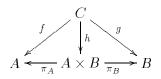
Definition A.2.2 Let C be a category. An object 1 is a **terminal** object in C, if for all C-objects A, there is only one morphism $f \in C(A, 1)$.

Remark A.2.3 For each object $A \in \mathbb{C}$, $\mathbf{t}_A : A \to 1$ denotes the unique morphism in \mathbb{C} from A to 1.

Definition A.2.4 Let C be a category and 1 a terminal object. A C-morphism $f: 1 \to A$ is called a **point** (or **constant** of type A or global element of A). Moreover, an object A of a category C is called **non-empty** if there is such a morphism.

A.3 Products

Definition A.3.1 . A **product** of two objects A and B in a category \mathbb{C} , is a \mathbb{C} -object $A \times B$, together with two projection morphisms $\pi_A \in \mathbb{C}(A \times B, A)$ and $\pi_B \in \mathbb{C}(A \times B, B)$ such that, for any object C and a pair of morphisms $f \in \mathbb{C}(C, A)$ and $g \in \mathbb{C}(C, B)$, there exists exactly one morphism $h \in \mathbb{C}(C, A \times B)$ such that the following diagram commutes, i.e., such that $\pi_A \circ h = f$ and $\pi_B \circ h = g$.



Definition A.3.2 Let C be a category. If the product objects $A \times C$ and $B \times D$ exist, then for each pair of morphisms $f: A \to B$, and $g: C \to D$, the **product morphism** $f \times g: A \times C \to B \times D$ is the morphism $\langle f \circ \pi_A, g \circ \pi_C \rangle : A \times C \to B \times D$.

Example A.3.3 In a poset (P, \leq_P) considered as a category, for $p, q \in P$, the product of p and q when it exists, is defined by the properties:

- $p \times q \leq p$, $p \times q \leq q$, i.e., $p \times q$ is a lower bound of p and q.
- If $c then <math>c , i.e., <math>p \times q$ is the greatest lower bound (g.l.b) of p and q.

Here, the fact that $c \leq_P$ is unique is not important, since in a poset there exists at most one morphism between two objects.

Proposition A.3.4 As a consequence of the definition of a product we have that $\langle \pi_A \circ h, \pi_B \circ h \rangle = h$.

Proposition A.3.5 $(f \times h) \circ (g \times k) = (f \circ g) \times (h \circ k)$

Proposition A.3.6 $\langle f \circ h, g \circ h \rangle = \langle f, g \rangle \circ h$.

Proposition A.3.7 $id_A \times id_B = id_{A \times B}$.

Proposition A.3.8

- 1. $(f \times h) \circ \langle g, k \rangle = \langle f \circ g, h \circ k \rangle$.
- 2. $\langle g, k \rangle = \langle g \times id \rangle \circ \langle id, k \rangle$.

A.4 Functors

Definition A.4.1 Let \mathbf{C} and \mathbf{D} be categories. A covariant functor $F: \mathbf{C} \to \mathbf{D}$ is a pair of mappings, $F_{Ob}: Ob(\mathbf{C}) \to Ob(\mathbf{D})$, $F_{Mor}: Mor(\mathbf{C}) \to Mor(\mathbf{D})$ for which

- 1. If $f: A \to B$ in \mathbb{C} , then $F_{Mor}(f): F_{Ob}(A) \to F_{Ob}(B)$ in \mathbb{D} .
- 2. $F_{Mor}(id_A) = id_{F_{Ob}(A)};$
- 3. $F_{Mor}(g \circ f) = F_{Mor}(g) \circ F_{Mor}(f)$.

Definition A.4.2 Let C and D be categories. A contravariant functor $F: C \to D$ is a pair of mappings $F_{Ob}: Ob(C) \to Ob(D)$, $F_{Mor}: Mor(C) \to Mor(D)$ for which

- 1. If $f: A \to B$ in \mathbf{C} , then $F_{Mor}(f): F_{Ob}(B) \to F_{Ob}(A)$ in \mathbf{D} .
- 2. $F_{Mor}(id_A) = id_{F_{Ob}(A)};$
- 3. $F_{Mor}(g \circ f) = F_{Mor}(f) \circ F_{Mor}(g)$.

Remarks A.4.3

1. Informally, a contravariant functor is a functor which reverses the direction of morphisms and hence, also the order of composition.

2. It is usual practice to omit the subscripts "Ob" and "Mor" as it always clear from context whether the functor is meant to operate on objects or on morphisms.

Example A.4.4 The **duality** functor $(-)_{\mathbf{C}}^{op}: \mathbf{C} \to \mathbf{C}^{op}$ is a contravariant functor such that for each $A \in \mathbf{C}$, $(A)^{op} = A$ and for each $f: A \to B \in \mathbf{C}$, $(f: A \to B)^{op} = f^{op}: B \to A$. We omit subscripts in the duality functor whenever this is clear from context. Note that we also have the inverse $(-)_{\mathbf{C}}^{op^{-1}}: \mathbf{C}^{op} \to \mathbf{C}$ such that $(-)^{op} \circ (-)^{op^{-1}} = I_{\mathbf{C}^{op}}$ and $(-)^{op^{-1}} \circ (-)^{op} = I_{\mathbf{C}}$.

Example A.4.5 (The dual functor) If $F: \mathbf{A} \to \mathbf{B}$ is a functor, its object function $A \mapsto F(A)$ and its mapping function $f \mapsto F(f)$, rewriten as $f^{op} \mapsto (F(f))_{\mathbf{B}}^{op} = ((-)_{\mathbf{B}}^{op} \circ F)(f)$, together define a functor from \mathbf{A}^{op} to \mathbf{B}^{op} , which we denote as $F^{op}: \mathbf{A}^{op} \to \mathbf{B}^{op}$ Note that $(F^{op})^{op^{-1}} = F$ since we have:

$$(F^{op})^{op^{-1}} = (-)^{op^{-1}} \circ F^{op}$$
$$= (-)^{op^{-1}} \circ (-)^{op} \circ F$$
$$= I_{\mathbf{C}} \circ F$$
$$= F$$

Example A.4.6 Consider now a functor $F: \mathbf{C^{op}} \to \mathbf{D}$. By definition of a functor, it assigns to each object $C \in \mathbf{C^{op}}$ a object F(C) in \mathbf{D} , and to each morphism $f^{op}: B \to A \in \mathbf{C^{op}}$ a morphism $F(f^{op}): F(B) \to F(A)$ in \mathbf{D} , with $F(f^{op} \circ g^{op}) = F(f^{op}) \circ F(g^{op})$ whenever $f^{op} \circ g^{op}$ is defined in $\mathbf{C^{op}}$ (i.e, when $g \circ f$ is defined in \mathbf{C}). The functor so described may be expressed directly in terms of the original category \mathbf{C} as a functor $F': \mathbf{C} \to \mathbf{D}$ as follows:

$$F' = F \circ (-)^{op}$$

so that for each $f: A \to B$ in \mathbb{C}

$$F'(f:A \to B) = (F \circ (-)^{op})(f:A \to B)$$

$$= F((-)^{op}(f))$$

$$= F(f^{op}:B \to A)$$

$$= F(f^{op}):F(B) \to F(A)$$

$$= F \circ (-)^{op}(f):F \circ (-)^{op}(B) \circ F \circ (-)^{op}(A)$$

$$= F'(f):F'(B) \to F'(A)$$

To see that $F': \mathbf{C} \to \mathbf{D}$ preserve identities and compositions note that

$$F'(id_A) = F \circ (-)^{op}(id_A)$$

$$= F(id_A^{op})$$

$$= id_{F(A)}$$

$$= id_{F \circ (-)^{op}(A)}$$

$$= id_{F'(A)}$$

while for each $f: A \to B, g: B \to C \in \mathbf{C}$

$$\begin{array}{lll} F'(g \circ f) & = & (F \circ (-)^{op})(g \circ f) \\ & = & F \circ ((g \circ f)^{op}) \\ & = & F(f^{op} \circ g^{op}) \\ & = & F(f^{op}) \circ F(g^{op}) \\ & = & F \circ (-)^{op}(f) \circ F \circ (-)^{op}(g) \\ & = & F'(f) \circ F'(g) \end{array}$$

Remark A.4.7 An immediate consequence of the above discussion is that any contravariant functor $F: \mathbf{C} \to \mathbf{D}$ me be expressed covariantly as $F: \mathbf{C}^{op} \to \mathbf{D}$ and vice-versa.

Example A.4.8 For every category C, we have the identity functor I_C which takes each C-object and every C-morphism to itself. More precisely we have that $I_C(g \circ f) = g \circ f = I_C(g) \circ I_C(f)$, and $I_C(id_A) = id_A = id_{I_C(A)}$.

Example A.4.9 Let \mathbf{C} be a category with all binary products. Then each \mathbf{C} -object A determines a functor $(-\times A): \mathbf{C} \to \mathbf{C}$, which takes each object B to $B \times A$, and each morphism $f: B \to C$ to $f \times id_A: B \times A \to C \times A$. The "-" is used to show where the argument object or morphism goes. A computer scientist might write $(-\times A)$ as $\lambda x.x \times A$. This functor is called the **right product functor**. Similarly, there is the **left product functor** $(A \times -): \mathbf{C} \to \mathbf{C}$.

Example A.4.10 The diagonal funtor $\Delta : \mathbf{C} \to \mathbf{C} \times \mathbf{C}$ takes each **C**-object A to the object $\langle A, A \rangle$ in the category $\mathbf{C} \times \mathbf{C}$, and each **C**-morphism $f : A \to B$ to the morphism $\langle f, f \rangle$ in the category $\mathbf{C} \times \mathbf{C}$.

Example A.4.11 Let \mathbf{C} be a category. Each \mathbf{C} -object A determines a functor $Hom_{\mathbf{C}}(A,-)$: $\mathbf{C} \to \mathbf{Set}$ (also denoted in the literaure of category theory as $\mathbf{C}(A,-): \mathbf{C} \to \mathbf{Set}$). This functor takes each \mathbf{C} -object B to the set $Hom_{\mathbf{C}}(A,B)$ of morphisms from A to B and each \mathbf{C} -morphism $f: B \to C$ to the function $Hom_{\mathbf{C}}(A,f): Hom_{\mathbf{C}}(A,B) \to Hom_{\mathbf{C}}(A,C)$ defined for each $g: A \to B$ by:

$$Hom_{\mathbf{C}}(A, f)(q: A \to B) = f \circ q.$$

 $Hom_{\mathbf{C}}(A, B)$ is called a **hom-functor**. The set $Hom_{\mathbf{C}}(A, B)$ is often called a **hom-set**. On the other hand, note that, for this definition to make sense, \mathbf{C} must be a locally small category (see A.1.9).

Example A.4.12 For a given object D, there's another covariant hom-functor $Hom_{\mathbf{C}}(-,D)$: $\mathbf{C}^{op} \to \mathbf{Set}$ (or equivalently, contravariant hom-functor $Hom_{\mathbf{C}}(-,D)$: $\mathbf{C} \to \mathbf{Set}$, see A.4.6) which is defined for each object A by $Hom_{\mathbf{C}}(-,D)(A) = Hom_{\mathbf{C}}(A,D)$ and for each morphism $f^{op}: B \to A$ in \mathbf{C}^{op}) by

$$Hom_{\mathbf{C}}(-,D)(f^{op}) = Hom_{\mathbf{C}}(f^{op},D) : Hom_{\mathbf{C}}(B,D) \to Hom_{\mathbf{C}}(A,D)$$

such that for each $k: B \to D, Hom_{\mathbf{C}}(f^{op}, D)(k) = k \circ f$.

Example A.4.13 The two variable hom-functor $Hom_{\mathbf{C}}(-,-): \mathbf{C}^{op} \times \mathbf{C} \to \mathbf{Set} (\mathbf{C}^{op} \times \mathbf{C})$ is the product category in the sense of A.1.5) takes a pair $\langle C, D \rangle$ of objects such that we have

$$Hom_{\mathbf{C}}(-,-)(\langle C,D\rangle) = Hom_{\mathbf{C}}(C,D)$$

and a pair of morphisms $\langle f^{op} : A \to C, g : B \to D \rangle \in \mathbf{C}^{op} \times \mathbf{C}$, (and hence $f : C \to A \in \mathbf{C}$), such that

$$Hom_{\mathbf{C}}(-,-)\langle f^{op}, g \rangle = Hom_{\mathbf{C}}(f^{op}, g)Hom_{\mathbf{C}}(A, B) \to Hom_{\mathbf{C}}(C, D)$$

Now, for any $h: A \to B$, $Hom_{\mathbf{C}}\langle f^{op}, g \rangle(h) = g \circ h \circ f$. Note that this composition really defines a morphism from C to D.

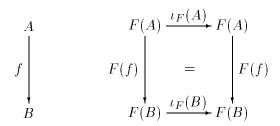
Proposition A.4.14 If $F : \mathbf{A} \to \mathbf{B}$ and $G : \mathbf{B} \to \mathbf{C}$ are functors, then the **composition** $(G \circ F) : \mathbf{A} \to \mathbf{C}$ defined by $(G \circ F)(A) = G(F(A))$ and $(G \circ F)(f) = G(F(f))$ is also a functor.

A.5 Natural transformations

Definition A.5.1 Let \mathbf{C} and \mathbf{D} be categories and F and G be functors from \mathbf{C} to \mathbf{D} . A natural transformation η from F to G, written $\eta: F \to G$, is a function that assigns to every \mathbf{C} -object A a \mathbf{D} -morphism $\eta_A: F(A) \to G(A)$ such that for any \mathbf{C} -arrow $f: A \to B$ the diagram on the right commutes in \mathbf{D} :

$$\begin{array}{ccc}
A & F(A) \xrightarrow{\eta_A} G(A) \\
f \downarrow & F(f) \downarrow & = \downarrow G(f) \\
B & F(B) \xrightarrow{\eta_B} G(B)
\end{array}$$

Example A.5.2 For every functor $F: \mathbf{C} \to \mathbf{D}$, the components of the identity natural transformation $\iota_F: F \to F$ are the identity morphisms of the objects in the image of F, i.e., $\iota_F(A) = id_{F(A)}$. To see this, note that the following diagram commutes for every morphism $f: A \to B$, i.e. $F(f) \circ \iota_F(A) = F(f) \circ id_{F(A)} = F(f) = id_{F(B)} \circ F(f) = \iota_F(B) \circ F(f)$.



Definition A.5.3 Let F, G be functors from C to D. If each component η_A of $\eta : F \to G$ is an isomorphism in D, we call η a natural isomorphism.

Remark A.5.4 In case η is a natural isomorphism, we can interpret this as meaning that the F-picture and the G-picture of \mathbf{C} look the same in \mathbf{D} . Each $\eta_A:F(A)\to G(A)$ then has an inverse $\eta_A^{-1}:G(A)\to F(A)$, and each such inverse is the component of a natural isomorphism $\eta^{-1}:G\to F$ (proposition A.5.5).

Proposition A.5.5 Suppose $F: \mathbf{C} \to \mathbf{D}$ and $G: \mathbf{C} \to \mathbf{D}$ are functors and $\eta: F \to G$ is natural isomorphism. Then there is a unique natural transformation $\eta^{-1}: G \to F$ such that the composites $\eta \circ \eta^{-1} = \iota_G$ and $\eta^{-1} \circ \eta = \iota_F$.

References

- [1] ADAMEK, Jiri; HERRLICH, Horst; STRECKER, Gorge E. Abstract and Concrete Categories: The Joy of Cats. New York: John Wiley & Sons, 1990. 482p.
- [2] ASPERTI, Andrea; LONGO, Giuseppe. Categories, Types, and Structures: An Introduction to Category Theory for the Working Computer Scientist. Cambridge: Mit Press, 1991. 306p.
- [3] ARBIB, Michael A; MANES, Ernest G. Algebraic Approaches to Program Semantics. New York: Springer-Verlag, 1986. 351p.
- [4] BARENDREGT, H. P. **The Lambda Calculus:** Its Syntax and Semantics. Amsterdam: North Holland, 1984. 621p.
- [5] BARR, M.; WELLS, C. Category Theory for Computing Science. New York: Prentice-Hall, 1990. 432p.
- [6] CROLE, R. L. Category for Types. Cambridge: Cambridge University Press, 1993, 335.
- [7] CARDELLI, Luca; WEGNER, Peter. On Understanding Types, Data Abstraction, and Polymorphism. Computing Surveys, New York, v.17, n.4, p.471-522, Dec. 1985.
- [8] Ehrig, Hartmut; Mahr; Bernd. Fundamentals of Algebraic Specification I: EATCS Monographs on Theoretical Computer Science, v.6. Berlin: Springer-Verlag, 1985. 321p.
- [9] GOLDBLATT, Robert. **TOPOI: The Categorial Analysis of Logic**: Studies in logic and the foundations of mathematics. New York: North-Holland, v.98, 1983. 551p.
- [10] GUNTER, Carl A. Semantics of Programming Languages: Structures and Techniques. Cambridge: Mit Press, 1992. 419p.
- [11] HOPCROFT, John; ULMANN, Jeffrey D. Introduction to Automata Theory, Languages, and Computation. Reading: Addison Wesley, 1979, 418p.
- [12] LANDIN, P. A correspondence between ALGOL 60 and Church's lambda notation. Communications of the ACM, New York, v.8 n.2, p. 80-101, Feb. 1965.
- [13] LAMBEK, J.; SCOTT, P. J. Introduction to Higher Order Categorical Logic. Cambridge: Cambridge University Press, 1986. 293p.
- [14] MaC LANE, Saunders. Categories for the Working Mathematician. New York: Springer-Verlag, 1971. 262p.
- [15] MARTINI, Alfio; EHRIG, H; NUNES, D. Elements of Basic Category Theory. TU Berlin, Technical Report 96-5, FB Informatik, 1996.
- [16] MITCHELL, John C. Type Systems for Programming Languages. In: **Handbook of Theoretical Computer Science**: Formal Models and Semantics.Cambridge: Mit Press, 1990, v.B, p.366-458.

- [17] PIERCE, Benjamin. A Taste of Category Theory for Computer Scientists. Pittsburgh: Carnegie Mellon University, CMU-CS-90-113, 1990. 86p.
- [18] POIGNÉ, Axel. Cartesian Closure–Higher Types in Categories. In: Category Theory and Computer Programming: Tutorial and Workshop, 9., 1985, Guildford. Lecture Notes in Computer Science, v.240. Berlin: Springer-Verlag, 1985, 519p. p.58-75.
- [19] THOMPSON, Simon. **Type Theory and Functional Programming**. Reading: Addison Wesley, 1991. 372p.
- [20] WYNSKEL, Glynn. **The Formal Semantics of Programming Languages:** An Introduction. Cambridge: MIT Press, 1993. 361p.