Introduction

In the following I attempted to give an account of some parts of semantics of typed lambda calculi. The work is divided into two parts. In the first part I briefly study syntax and categorical semantics of simply typed lambda calculus. In the second part I then concentrate on second order lambda calculus. In chapter 3 I give a definition of second order lambda calculus and discuss some aspects of its syntax. In chapter 4 I give a general definition of a model of second order lambda calculus and briefly describe two concrete models. In the fifth chapter I give a detailed description of an important model of second order lambda calculus based on Scott domains.

My main goal was to describe ideas which led to the definition of second order lambda calculus and to present several different approaches to modelling second order lambda calculus. I paid special attention to showing some connections between these approaches.

The work was derived from the papers cited in References. Sometimes, when I thought it useful, I supplied some details, proofs and examples.

This paper was written during my stay at Brno University as a part of my Mgr. course in discrete mathematics. I would like to thank prof. J. Rosický for supervising my study at that time.

0 Categories

Before we start, it will be better to review some necessary notions from category theory, mainly in order to standartize notation.

Definition. Let $\mathbf{C}$ be a category, $A, B \in \mathbf{C}$. The product of A and B is an object $A \times_{\mathbf{C}} B$ (where the index is clear from context we will omit it) together with projections

$$\mathbf{fst}_{A,B} \colon A \times B \to A$$
$$\mathbf{snd}_{A,B} \colon A \times B \to B$$

such that, for any $C \in \mathbf{C}$ and any pair of morphisms $f \colon C \to A$ and $g \colon C \to B$, there is a unique morphism $h \colon C \to A \times B$ such that

$$\mathbf{fst} \circ h = f$$
$$\mathbf{snd} \circ h = g.$$

Remark. Given morphisms $f \colon A_1 \to A_2$, $g \colon B_1 \to B_2$ such that $A_1 \times B_1$ and $A_2 \times B_2$ exist. Then $f \times g$ denotes the uniquely determined morphism from $A_1 \times B_1$ to $A_2 \times B_2$ which makes the following diagram commute

$$\begin{array}{ccc} A_1 & \xrightarrow{f} & A_2 \\ {\scriptstyle \mathbf{fst}}\uparrow & & \uparrow{\scriptstyle \mathbf{fst}} \\ A_1 \times B_1 & \xrightarrow{f \times g} & A_2 \times B_2. \\ {\scriptstyle \mathbf{snd}}\downarrow & & \downarrow{\scriptstyle \mathbf{snd}} \\ B_1 & \xrightarrow[g]{} & B_2 \end{array}$$

Definition. A cartesian closed category $\mathbf{C}$ is a category satisfying the following:

(1) There is a terminal object $1_{\mathbf{C}}$.
(2) For each pair of objects A and B of $\mathbf{C}$ there is a product $A \times_{\mathbf{C}} B$ with projections $\mathbf{fst}_{A,B} \colon A \times_{\mathbf{C}} B \to A$ and $\mathbf{snd}_{A,B} \colon A \times_{\mathbf{C}} B \to B$.
(3) For each pair of objects A and B of $\mathbf{C}$ there is an exponent $A \to_{\mathbf{C}} B$ together with morphism $\mathrm{eval} \colon (A \to_{\mathbf{C}} B) \times_{\mathbf{C}} A \to B$ such that, for any morphism $f \colon C \times_{\mathbf{C}} A \to B$, there is a morphism $\lambda f \colon C \to (A \to_{\mathbf{C}} B)$ such that

$$\mathrm{eval} \circ (\lambda f \times \mathbf{id}_A) = f.$$

Remarks.

(1) We can look at λ as an operation $\lambda\colon \mathrm{Hom}_{\mathbf{C}}(C \times_{\mathbf{C}} A, B) \to \mathrm{Hom}_{\mathbf{C}}(C, A \to_{\mathbf{S}} B)$ (it would be more correct to write $\lambda_{\mathbf{C}}^{A,B}$ but we will omit the indexes where possible) which is an isomorphism natural in C.

(2) An important point is that we have a **given** terminal object, product and exponent for every pair of objects. There are two slightly different notions of a functor preserving the cartesian structure. If functor $F\colon \mathbf{C} \to \mathbf{C}'$ preserves terminal objects, binary products and exponents (i.e. the image of a terminal object in $\mathbf{C}$ is a terminal object in $\mathbf{C}'$ and whenever we have a product diagram

$$A \xleftarrow{p} P \xrightarrow{q} B$$

in the category $\mathbf{C}$, the diagram

$$F(A) \xleftarrow{F(p)} F(P) \xrightarrow{F(q)} F(B)$$

is a product diagram in $\mathbf{C}'$, similarly for exponents) we say that F preserves the cartesian closed structure. If moreover the image of the given terminal object $1_{\mathbf{C}}$ in $\mathbf{C}$ is the given $1_{\mathbf{C}'}$ in $\mathbf{C}'$, $F(A \times_{\mathbf{C}} B) = F(A) \times_{\mathbf{C}'} F(B)$ and $F(A \to_{\mathbf{C}} B) = F(A) \to_{\mathbf{C}'} F(B)$ then we say that F preserves the cartesian closed structure on the nose (sometimes we say that F is strict).

(3) Sometimes the exponent is defined by means of right adjoints to the functors $(-) \times_{\mathbf{C}} A\colon \mathbf{C} \to \mathbf{C}$ for every $A \in \mathbf{C}$. Indeed from our definition it follows that in a cartesian closed category this functor has a right adjoint (usually denoted $(-)^A\colon \mathbf{C} \to \mathbf{C}$ such that $(B)^A = A \to_{\mathbf{C}} B$.

Definition. Given two categories $\mathbf{C}$ and $\mathbf{C}'$. The product category $\mathbf{C} \times \mathbf{C}'$ is defined to be the category which has as objects pairs (A, A') where $A \in \mathbf{C}$ and $A' \in \mathbf{C}'$. The morphisms are pairs $(f, g)\colon (a, a') \to (B, B')$ where $f\colon A \to B$ and $g\colon A' \to B'$.

Remarks.

(1) We have also projections

$$\mathbf{Fst}_{\mathbf{C},\mathbf{C}'}\colon \mathbf{C} \times \mathbf{C}' \to \mathbf{C}$$
$$\mathbf{Snd}_{\mathbf{C},\mathbf{C}'}\colon \mathbf{C} \times \mathbf{C}' \to \mathbf{C}'.$$

(Usually we will write **Fst** and **Snd** for projection functors and **fst** and **snd** for projection morphisms inside a category).

(2) For $1 \leq i \leq m$, we put $\mathbf{P}^{i,m}_{\mathbf{C}_1,\ldots,\mathbf{C}_n}\colon \mathbf{C}_1 \times \cdots \times \mathbf{C}_m \to \mathbf{C}_i$ to be the ith projection.

(3) Given functors $F\colon \mathbf{C} \to \mathbf{C}_1$ and $G\colon \mathbf{C} \to \mathbf{C}_2$, $< F, G >$ is then the unique functor from $\mathbf{C}$ to $\mathbf{C}_1 \times \mathbf{C}_2$ such that $\mathbf{Fst} \circ < F, G >= F$ and $\mathbf{Snd} \circ < F, G >= G$.

(4) If $F\colon \mathbf{C} \to \mathbf{D}$ and $G\colon \mathbf{C}' \to \mathbf{D}'$ are functors then we define

$$F \times G =< F \circ \mathbf{Fst}, G \circ \mathbf{Snd} >\colon \mathbf{C} \times \mathbf{C}' \to \mathbf{D} \times \mathbf{D}'.$$

(5) We sometimes write $\mathbf{1}$ for the category with one object and one arrow (i.e. for the terminal object in the category **Cat** – the category of small categories and functors between them).

Definition. Let $\mathbf{C}$ and $\mathbf{D}$ be categories. An adjunction from $\mathbf{D}$ to $\mathbf{C}$ is a triple $< F, G, \varphi >$ such that

(1) $F : \mathbf{D} \to \mathbf{C}$ and $G : \mathbf{C} \to \mathbf{D}$ are functors (F is called left adjoint of G and G is called right adjoint of F).
(2) $\varphi : \mathrm{Hom}_{\mathbf{C}}(F-,-) \cong \mathrm{Hom}_{\mathbf{D}}(-,G-)$ is a natural isomorphism of hom functors.

Remarks.

(1) So φ is a family of mappings indexed by pairs (D, C) where $D \in \mathbf{D}$ and $C \in \mathbf{C}$ such that $\varphi_{D,C} : \mathrm{Hom}_{\mathbf{C}}(F(D), C) \to \mathrm{Hom}_{\mathbf{D}}(D, G(C))$ is isomorphism. Moreover, this isomorphism is natural in D and C, that is, for any $f : F(D) \to C$, $k : C \to C'$ and $h : D' \to D$, we have

$$\varphi_{D,C'}(k \circ f) = G(k) \circ \varphi_{D,C}(f)$$
$$\varphi_{D',C}(f \circ F(h)) = \varphi_{D,C}(f) \circ h.$$

(2) The natural transformation $\eta : \mathbf{Id}_{\mathbf{D}} \to G \circ F$ defined by

$$\eta_D = \varphi_{D,F(D)}(\mathbf{id}_{F(D)}) : D \to G(F(D))$$

for $D \in \mathbf{D}$ is called the unit of the adjunction and the natural transformation $\epsilon : F \circ G \to \mathbf{Id}_{\mathbf{C}}$ defined by

$$\epsilon_C = \varphi^{-1}_{G(C),C}(\mathbf{id}_{G(C)} : F(G(C)) \to C$$

for $C \in \mathbf{C}$ is called the counit of the adjunction.

PART I.
Simply Typed Lambda Calculus

A simply typed lambda calculus is at the basis of all typed systems. Introduced by Church in 1940 as an attempt to avoid Russell's paradox, it has become one of the most important concepts in theoretical computer science as well as in logic. In this part we give a definition of a simply typed lambda calculus, discuss briefly its syntax and categorical semantics. I concentrated in this part on describing those features which help to understand the discussion about second order lambda calculus studied in part II.

1. Syntax

First we define what the terms and the types of a simply typed lambda calculus are.

Definition. The types of a simply typed lambda calculus (over a ground set At) are given inductively as follows:

(1) every member of At is a type.
(2) If σ and τ are types, then $\sigma \to \tau$ and $\sigma \times \tau$ are types.

Remarks.

(1) We shall use small greek letters to denote types.
(2) The intuitive meaning of operators $\to$ and $\times$ is clear: $\sigma \to \tau$ denotes the function type and $\sigma \times \tau$ the product type.
(3) Some authors mean by simply typed lambda calculus a typed system without the product operator. A typed system with types given as above is then called a simply typed lambda calculus with explicit pairs.

Definition. The terms of a simply typed lambda calculus are those that can be generated by the following clauses:

(1) for every type σ, there is a countable set $\{x_1, x_2, \dots\}$ variables of type σ. We shall write $x : \sigma$.
(2) if $x : \sigma$ is a variable and $M : \tau$ is a term, then $\lambda x : \sigma . M$ is a term of type $\sigma \to \tau$. We

say that x is bounded in $\lambda x : \sigma.M$.

(3) if $M : \sigma \to \tau$ and $N : \sigma$ are terms, then $M(N)$ is a term of type τ.

(4) if $M : \sigma$ and $N : \tau$ are terms, then $< M, N >$ is a term of type $\sigma \times \tau$.

(5) if $M : \sigma \times \tau$ is a term, then $\mathrm{fst}(M)$ and $\mathrm{snd}(M)$ are terms of types σ and τ respectively.

Now we proceed to define the equational theory of simply typed lambda calculus.

Definition. The equational theory of simply typed lambda calculus is defined to be the minimal congruence relation "=" satisfying the following axiom schemas:

$$\begin{aligned}
\alpha : &\quad \lambda x : \sigma.M = \lambda y : \sigma.[y/x]M \\
\to \beta : &\quad (\lambda x : \sigma.M)N = [N/x]M \\
\to \eta : &\quad \lambda x : \sigma.(Mx) = M \\
\times\beta_1 : &\quad \mathrm{fst}(< M, N >) = M \\
\times\beta_2 : &\quad \mathrm{snd}(< M, N >) = M \\
\times\eta : &\quad < \mathrm{fst}(M), \mathrm{snd}(M) >= M
\end{aligned}$$

where the types of the terms are such that the terms are correctly formed. Moreover, in the α rule y is not free in M and in the $\to \beta$ rule x is not free in M.

Remark. Instead of working with equations between terms we could define the notion of reduction. The previous equations then would become rewriting rules. The immediate reductions are

$$\begin{aligned}
\to \beta : &\quad (\lambda x : \sigma.M)N \hookrightarrow [N/x]M \\
\to \eta : &\quad \lambda x : \sigma.(Mx) \hookrightarrow M \\
\times\beta_1 : &\quad \mathrm{fst}(< M, N >) \hookrightarrow M \\
\times\beta_2 : &\quad \mathrm{snd}(< M, N >) \hookrightarrow M \\
\times\eta : &\quad < \mathrm{fst}(M), \mathrm{snd}(M) > \hookrightarrow M
\end{aligned}$$

(again in the $\to \beta$ rule x is not free in M). We then define the reduction to be the smallest transitive relation containing $\hookrightarrow$ and compatible with the formation of terms. We write $M \hookrightarrow^* N$ when M reduces to N. We say that a term M is normal if no immediate reduction can be applied on any of its subterms. A term M is strongly normalisable when there is no infinite reduction sequence starting with M.

Theorem. (1.1) *All terms of a simply typed lambda calculus are strongly normalisable and the normal form is unique.*

Proof. Can be found in [**Girard et al. 89**]. □

Remark. The type theory presented here has its logical counterpart. The types can be regarded as formulas of (intuitionistic) first-order propositional logic (based on connectives $\wedge$ and $\Rightarrow$). The terms then denote deductions of their types, under hypotheses which are the types of their free variables. The reduction rules correspond to the eliminations of unnecessary "detours" in the deductions. This correspondence between deductions in

natural deduction and terms of simply typed lambda calculus is known as the Curry-Howard "formulae-as-types" isomorphism.

The question which now comes to mind is what the expressive power of this system is. For example, is it possible to interpret in this system the type of integers, and if it is so, what functions between integers can be expressed? Before we answer this questions, we state the following lemma.

Lemma. (1.2) *The only normal terms of type $\sigma \to \tau$ and $\sigma \times \tau$ are of the forms $\lambda x : \sigma.N$ and $< N_1, N_2 >$ respectively.*

PROOF. We prove both parts of the lemma simultaneously by induction on the structure of term M. There are five cases.

(1) $M \equiv \lambda x : \sigma_1.M_1$.
  (i) Suppose that M is a normal term of type $\sigma \to \tau$. It follows that $\sigma_1 = \sigma$ and M is of the required form.
  (ii) The case that M is a normal term of type $\sigma \times \tau$ is immediately ruled out since the types are incompatible.

(2) $M \equiv M_1(M_2)$.
  (i) M of type $\sigma \to \tau$. Then M_1 must have type $\sigma_1 \to (\sigma \to \tau)$ for some type σ_1. By induction hypothesis $M_1 = \lambda x_1 : \sigma_1.M_1'$ and so $M_1(M_2)$ is not a normal term.
  (ii) M of type $\sigma \times \tau$. Then M_1 must have type $\sigma_1 \to (\sigma \times \tau)$. Again by induction hypothesis, $M_1 = \lambda x_1 : \sigma_1.M_1'$ and so $M_1(M_2)$ is not a normal term.

(3) $M \equiv< M_1, M_2 >$.
  (i) M of type $\sigma \to \tau$. This cannot happen since the types are incompatible.
  (ii) M of type $\sigma \times \tau$. It follows that $\sigma_1 = \sigma$ and so M is of the required form.

(4) $M \equiv \mathrm{fst}(M_1)$.
  (i) Then M_1 must have type $(\sigma \to \tau) \times \sigma_1$ and again by the induction hypothesis the term $\mathrm{fst}(M_1)$ is not normal.
  (ii) Then M_1 is of the type $(\sigma \times \tau) \times \sigma_1$. By the induction hypotheses M is not a normal term.

(5) $M \equiv \mathrm{snd}(M_1)$. The same argument as in the previous case applies. □

One way to represent integers in simply typed lambda calculus is to represent them as the closed normal terms of type

$$\mathtt{Int}_\sigma = (\sigma \to \sigma) \to (\sigma \to \sigma)$$

where σ is a type. Term

$$n_\sigma = \lambda y\colon \sigma \to \sigma.\lambda x : \sigma.\underbrace{y(y \ldots (y}_{n \text{ times}}(x)) \ldots)$$

then represents the natural number n. We'll show that this representation is, at least to some extent, satisfactory.

Lemma. (1.3) *Terms n_σ for some $n \in \mathbf{N}$ and the identity term $\lambda y\colon \sigma \to \sigma.y$ are the only closed normal terms of type $\mathtt{Int}_\sigma$.*

PROOF. Take M to be a closed normal term of type $\mathtt{Int}_\sigma$. From lemma 1.2 it follows that M is of the form $\lambda y\colon \sigma \to \sigma.M_1$, where M_1 is a normal term of type $\sigma \to \sigma$. M_1 is either a variable (y and M is then the identity term) or is of the form $\lambda x : \sigma.N$, where N is a normal term of type σ (extension of the argument in lemma 1.2). Now N is either a variable (and so x) or of the form $N_1(N_2)$ (the other cases are immediately ruled out). N_1 is a normal term of type $\tau \to \sigma$. But since N is normal, N_1 cannot be of the form $\lambda z\colon \tau.N_1'$ and so N_1 has to be a variable, and hence N_1 is x (and $\tau = \sigma$). N_2 is a normal term of type σ, so it is again either x or y applied to an expression of type σ. It follows that N is of the form $y(y \ldots (y(x)) \ldots)$ and so M is of the form n_σ for some $n \in \mathbf{N}$. $\square$

So it is possible to represent integers but what about functions on them? Given a closed term f of type $\mathtt{Int}_\sigma \to \mathtt{Int}_\sigma$, it induces a function $|f|$ from $\mathbf{N}$ to $\mathbf{N}$ defined as follows

$$|f|(n) = m \qquad \text{iff} \qquad f(\overline{n}) \hookrightarrow^* \overline{m}.$$

Since the normal form of every term exists and is unique, function $|f|$ is well defined and total.

Theorem. (1.4) *The functions from* $\mathbf{N}$ *to* $\mathbf{N}$ *representable in simply typed lambda calculus are those that can be generated from constants 0 and 1 using the operations addition, multiplication and conditional.*

PROOF. Can be found in [**Fortune et al. 83**]. $\square$

So we can see that the class of representable functions is not very big - we can represent only polynomials extended with the conditional function. So for a real work it is necessary to create richer typed systems - either by adding new constants and conversion rules (this is the case of Gödel's system T) or by allowing to perform more powerful operations on types. One of the typed systems we can get using the latter method is second order lambda calculus which we will study in Part II.

2 Categorical Semantics

The main idea of categorical semantics of a simply typed lambda calculus is to interpret types as objects and terms as morphisms in some category C. It isn't too difficult to see an analogy between the axioms of a simply typed lambda calculus and the axioms of a cartesian closed category. When we regard (a type) $A \times B$ as the categorical product of (types) A and B, then $< , >$ is the product morphism and fst( ) and snd( ) are the first and second projection respectively (application of terms now corresponds to composition of morphisms). The $\times\beta_1$, $\times\beta_2$ and $\times\eta$ are precisely the axioms defining a product in a cartesian closed category. A little bit trickier situation arises in the case of exponents - here we have to deal with the notion of substituion of a term for a free variable - something which doesn't have an immediate analogy in a category theory. But this problem can be overcome. Let's look a bit closer at our idea. What role do free variables play? A term $M : \sigma$, whose free variables are among $x_1 : \sigma_1, \ldots, x_n : \sigma_n$, is to be interpreted as a morphism from $\Delta = \sigma_1 \times \cdots \times \sigma_n$ to σ. Now if we have two terms $M : \sigma$ and $N : \tau$ such that free variables of $[N/x]M$ are among $x_1 : \sigma_1, \ldots, x_n : \sigma_n$, the substitution of N in M is modellled by composing M with $< \mathbf{id}_\Delta, N >$

$$\sigma_1 \times \cdots \times \sigma_n \xrightarrow{<\mathbf{id}_\Delta, N>} (\sigma_1 \times \cdots \times \sigma_n) \times \sigma \xrightarrow{M} \sigma.$$

The morphism we have got really has the required domain and codomain. The $\rightarrow \beta$ and $\rightarrow \eta$ rules are satisfied.

Before we summarize the previous discussion in a formal definition, we define the notion of a context.

Definition. A context H is a (possibly empty) list of variables, $H = (x_1 : \sigma_1, \ldots, x_n : \sigma_n)$. A term $M : \sigma$ is said to be legal in a context H (we shall write $H \vdash M : \sigma$) if every free variable of M is in H. We shall write $H, x : \sigma$ for a context $x_1 : \sigma_1, \ldots, x_n : \sigma_n, x : \sigma$.

Definition. Let C be a cartesian closed category and I a map assigning to each atomic type an object of C. Then the interpretation of simply typed lambda calculus in C is given as follows:

(1) A type σ is interpreted by object $[\![\sigma]\!]$ of C defined as follows:
  (i) $[\![\sigma]\!] = I(\sigma)$ for an atomic type σ.
  (ii) $[\![\sigma \rightarrow \tau]\!] = [\![\sigma]\!] \rightarrow_C [\![\tau]\!]$
  (iii) $[\![\sigma \times \tau]\!] = [\![\sigma]\!] \times_C [\![\tau]\!]$

(2) A term $M : \sigma$ legal in a context $H = (x_1 : \sigma_1, \ldots, x_n : \sigma_n)$ is interpreted by a morphism

$$[\![M]\!]_H : [\![\sigma_1]\!] \times_C \cdots \times_C [\![\sigma_n]\!] \rightarrow [\![\sigma]\!]$$

in C defined inductively as follows:
  (i) $[\![x_i]\!]_H = \mathbf{snd} \circ \mathbf{fst}^{n-i}$
  (ii) $[\![\lambda x : \tau . M]\!]_H = \lambda([\![M]\!]_{H,x:\tau})$
  (iii) $[\![M(N)]\!]_H = \mathrm{eval} \circ < [\![M]\!]_H, [\![N]\!]_H >$

(iv) $[\![< M, N >]\!]_H =< [\![M]\!]_H, [\![N]\!]_H >$
(v) $[\![\mathrm{fst}(M)]\!]_H = \mathbf{fst} \circ [\![M]\!]_H$
(vi) $[\![\mathrm{snd}(M)]\!]_H = \mathbf{snd} \circ [\![M]\!]_H$

Definition. We say that an equation $M = N$ where M and N are terms legal in context H is satisfied in the given interpretation if $[\![M]\!]_H = [\![N]\!]_H$.

From the next theorem it follows that the interpretation given above really makes sence.

Theorem. (2.1) (Soundness) *All equational rules for simply typed lambda calculus are valid under the interpretation given above.*

PROOF. A straightforward check of rules. □

So now we have three vertexes of a triangle - a simply typed lambda calculus, the corresponding propositional logic and its term model (forming a cartesian closed category). They represent three different approaches to the same notion. We shall meet with a similar situation in the case of second order lambda calculus and system λP which is the type theory of first order predicate logic.

PART II.
Second Order Lambda Calculus

A second order lambda calculus was first introduced by Girard in 1970 for the sake of proof theory of second order intuitionistic logic. Later it was independently rediscovered in computer science by Reynolds as a part of analysis of parametric polymorphism. Second order lambda calculus (system F called by Girard) arises as an extension of the simply typed lambda calculus, obtained by adding an operation of abstraction on types (universal abstraction). Consider for example a term $\lambda x : \alpha . x$ of a type $\alpha \to \alpha$. Since we made no assumption about the type α, we can regard α as a free type variable. Second order lambda calculus then allows us to construct term $\Lambda\alpha . \lambda x : \alpha . x$ which denotes the "polymorphic" identity function, that is a function which can be applied to any type σ and the result of the application is the identity function $\lambda x : \sigma . x$ on σ. The type of polymorphic identity $\prod \alpha . \alpha \to \alpha$ is obtained from the type $\alpha \to \alpha$ by means of universal abstraction $\prod$. So a term of a type $\prod \alpha . \sigma$ is a function which associates to every type τ an element of type $[\tau/\alpha]\sigma$. Now there is an obvious circularity problem (known as impredicativity of second order lambda calculus): this term can be applied to **any** type and so, in particular, also to its own type. One of the consequences of this circularity is a difficult modelling of the system. In the following chapter we shall be concerned with the syntax of second order lambda calculus and with its expressive power.

3 Syntax

Definition. The types of second order lambda calculus are those that can be generated by the following clauses:

(1) There is a countable set $\{\alpha_1, \alpha_2, \ldots\}$ of type variables.
(2) If σ and τ are types, then $\sigma \to \tau$ is a type.
(3) If σ is a type and α is a type variable, then $\prod \alpha . \sigma$ is a type. We say that variable α is bounded in $\prod \alpha . \sigma$.

Example. $\prod\alpha.\alpha$, $\prod\alpha.\alpha \to \alpha$, $\alpha \to \prod\alpha.\beta \to \alpha$ are examples of types.

Now we proceed to define the terms of the calculus. Since in the next chapters we will be interested mainly in the semantics of second order lambda calculus, we use in the definition the notion of a context.

Definition. A context Σ is a (possibly empty) list of type variables, $\Sigma = (\alpha_1, \ldots, \alpha_m)$. We shall write Σ, α for the context $\Sigma = (\alpha_1, \ldots, \alpha_m, \alpha)$. A type σ is legal in a context Σ if every free type variable of σ is in Σ.

Definition. A type assignment H legal in a context Σ is a list $H = (x_1 : \sigma_1, \ldots, x_n : \sigma_n)$ of typings for variables such that, for $1 \leq i \leq n$, σ_i is legal in Σ.

Definition. Terms of second order lambda calculus are those segments $H \vdash_\Sigma M : \sigma$ (where H is a type assignment legal in Σ and we say that M is of type σ under assignment H) derivable by the following typing rules:

projection: $\quad H_1, x : \sigma, H_2 \vdash_\Sigma x : \sigma$

$\to$ introduction: $\quad \dfrac{H, x : \sigma_1 \vdash_\Sigma M : \sigma_2}{H \vdash_\Sigma \lambda x : \sigma_1 . M : \sigma_1 \to \sigma_2}$

$\prod$ introduction: $\quad \dfrac{H \vdash_{\Sigma,\alpha} M : \sigma}{H \vdash_\Sigma \Lambda\alpha . M : \prod\alpha.\sigma}$

$\to$ elimination: $\quad \dfrac{H \vdash_\Sigma M : \sigma_1 \to \sigma_2, \qquad H \vdash_\Sigma N : \sigma_1}{H \vdash_\Sigma M(N) : \sigma_2}$

$\prod$ elimination: $\quad \dfrac{H \vdash_\Sigma M : \prod\alpha.\sigma}{H \vdash_\Sigma M\{\sigma_2\} : [\sigma_2/\alpha]\sigma_1}$.

These rules are subject to some restrictions:

(1) In the projection rule, the variable x does not appear in H_1 or H_2.
(2) In the $\prod$ introduction rule, there is no free occurrence of α in the type of any variable in H.
(3) In the $\prod$ elimination rule, all free variables of σ_2 are in Σ.

Remark. The second restriction ensures that, in a term $H \vdash_\Sigma \Lambda\alpha.M : \prod\alpha.\sigma$, the type assignment H is legal in Σ.

Example. $\Lambda\alpha.\lambda x : \alpha.x$ is a term of type $\prod\alpha.\alpha \to \alpha$. As we already mentioned, this term is called the polymorphic identity.

Now we define the equational rules of second order lambda calculus.

Definition. An equational rule of second order lambda calculus is an expression $H \vdash_\Sigma M = N : \sigma$ which is of one of the following forms:

reflexivity: $H_1, x:\sigma, H_2 \vdash_\Sigma x = x : \sigma$

ξ : $$\frac{H, x:\sigma_1 \vdash_\Sigma M = N : \sigma_2}{H \vdash_\Sigma \lambda x:\sigma_1.M = \lambda x:\sigma_1.N : \sigma_1 \to \sigma_2}$$

type ξ : $$\frac{H \vdash_{\Sigma,\alpha} M = N : \sigma}{H \vdash_\Sigma \Lambda\alpha.M = \Lambda\alpha.N : \prod\alpha.\sigma}$$

congruence: $$\frac{H \vdash_\Sigma M_1 = N_1 : \sigma_1, \qquad H \vdash_\Sigma M_2 = N_2 : \sigma_1 \to \sigma_2}{H \vdash_\Sigma M_2(M_1) = N_2(N_1) : \sigma_2}$$

type congruence: $$\frac{H \vdash_\Sigma M = N : \prod\alpha.\sigma_1}{H \vdash_\Sigma M\{\sigma_2\} = N\{\sigma_2\} : [\sigma_2/\alpha]\sigma_1}.$$

These rules imply that $H \vdash_\Sigma M : \sigma$ is term of second order lambda calculus if and only if $H \vdash_\Sigma M = M : \sigma$. Thus, in the remaining rules, we use $H \vdash_\Sigma M : \sigma$ for $H \vdash_\Sigma M = M : \sigma$.

symmetry: $$\frac{H \vdash_\Sigma M = N : \sigma}{H \vdash_\Sigma N = M : \sigma}$$

transitivity: $$\frac{H \vdash_\Sigma M = N : \sigma, \qquad H \vdash_\Sigma N = P : \sigma}{H \vdash_\Sigma M = P : \sigma}$$

β : $$\frac{H, x:\sigma_1 \vdash_\Sigma M : \sigma_2 \quad H \vdash_\Sigma N : \sigma_1}{H \vdash_\Sigma (\lambda x:\sigma_1.M)(N) = [N/x]M : \sigma_2}$$

type β : $$\frac{H \vdash_{\Sigma,\alpha} M : \sigma_1}{H \vdash_\Sigma (\Lambda\alpha.M)\{\sigma_2\} = [\sigma_2/\alpha]M : [\sigma_2/\alpha]\sigma_1}$$

η : $$\frac{H \vdash_\Sigma M : \sigma_1 \to \sigma_2}{H \vdash_\Sigma \lambda x:\sigma_1.M(x) = M : \sigma_1 \to \sigma_2}$$

type η : $$\frac{H \vdash_\Sigma M : \prod\alpha.\sigma}{H \vdash_\Sigma \Lambda\alpha.M\{\alpha\} = M : \prod\alpha.\sigma}.$$

These rules are again subject to certain restrictions:

(1) In the reflexivity rule, the variable x does not appear in H_1 or H_2.
(2) In the type ξ rule, there is no free occurrence of α in the type of a variable in H.
(3) In the type β rule, there is no free occurrence of α in the type of a variable in H.
(4) In the η rule, the variable x does not appear in H.
(5) In the type η rule, the variable α does not apppear in Σ.

Example. The type β rule tells us, for example, that

$$(\Lambda\alpha.\lambda x:\alpha.x)\{\sigma\} = \lambda x:\sigma.x,$$

i.e. that the value of the polymorphic identity on the type σ is indeed the identity function on σ.

Remark. We could again look at the given equations as at rewriting rules. The immediate reductions now are

$$\begin{array}{ll} \beta: & (\lambda x:\sigma.M)N \hookrightarrow [N/x]M \\ \eta: & \lambda x:\sigma.(Mx) \hookrightarrow M \\ \text{type } \beta: & (\Lambda\alpha.M)\{\sigma\} \hookrightarrow [\sigma/\alpha]M \\ \text{type } \eta: & \Lambda\alpha.M\{\alpha\} \hookrightarrow M \end{array}$$

(the restrictions given above apply). The reduction relation $\hookrightarrow^*$ is defined in the same way as in the case of simply typed lambda calculus . We have an analogy of theorem (1.1):

Theorem. (3.1) *All terms of second order lambda calculus are strongly normalisable and the normal form is unique.*

Proof. Can be found in [**Girard et al. 89**]. □

Remark. The Curry-Howard isomorphism for natural deduction can be extended for second order lambda calculus. The types are the formulas of (intuitionistic) second order propositional calculus and the terms denote deductions of their types, under hypotheses which are the types of their free variables. The typing rules $\prod$ introduction and $\prod$ elimination correspond to

$$\frac{\begin{array}{c}\vdots\\ \sigma\end{array}}{\forall\alpha.\sigma} \qquad\qquad \frac{\begin{array}{c}\vdots\\ \forall\alpha.\sigma\end{array}}{[\sigma_2/\alpha]\sigma_1},$$

respectively. The reduction rule $(\Lambda\alpha.M)\{\tau\} \hookrightarrow [\tau/\alpha]M$ corresponds to the conversion

$$\frac{\dfrac{\begin{array}{c}\vdots\\ \sigma\end{array}}{\forall\alpha.\sigma}}{[\tau/\alpha]\sigma} \quad \text{converts to} \quad \begin{array}{c}\vdots\\ [\tau/\alpha]\sigma.\end{array}$$

3.1 Representation of Types

First we state an analogy of lemma (1.2):

Lemma. (3.2) *The only normal terms of type $\sigma \to \tau$ and $\prod\alpha.\sigma$ are of the forms $\lambda x:\sigma.N$ and $\Lambda\alpha.N$ respectively.*

Proof. The proof is similar to the proof of lemma (1.2). We prove both parts of the lemma simultaneously by induction on the structure of the term M. There are four cases.

(1) $M \equiv \lambda x:\sigma_1.M_1$.

(i) Suppose that M is a normal term of type $\sigma \to \tau$. It follows that $\sigma_1 = \sigma$ and M is of the required form.
(ii) The case that M is a normal term of type $\prod \alpha.\sigma$ is immediately ruled out since the types are incompatible.

(2) $M \equiv \Lambda\alpha_1.M_1$.
(i) M of type $\sigma \to \tau$. This cannot happen since the type of M would be $\prod \alpha_1.\sigma_1 \neq \sigma \to \tau$.
(ii) M of type $\prod \alpha.\sigma$. It follows that $\alpha_1 = \alpha$ and so M is of the required form.

(3) $M \equiv M_1(M_2)$.
(i) M of type $\sigma \to \tau$. Then M_1 must have type $\sigma_1 \to (\sigma \to \tau)$. By induction hypothesis $M_1 = \lambda x_1 : \sigma_1.M_1'$ and so $M_1(M_2)$ is not a normal term.
(ii) M of type $\prod \alpha.\sigma$. Then M_1 must have type $\sigma_1 \to (\prod \alpha.\sigma)$. Again by induction hypothesis, $M_1 = \lambda x_1 : \sigma_1.M_1'$ and so $M_1(M_2)$ is not a normal term.

(4) $M \equiv M_1\{\sigma_1\}$. Then M_1 must have type $\prod \alpha_1.\sigma_1'$ and again by the induction hypothesis the term $M_1\{\sigma_1\}$ is not normal. □

Now we will show how to represent some basic types in second order lambda calculus.

Boolean

We define

$$\texttt{Bool} = \prod \alpha.\alpha \to (\alpha \to \alpha).$$

We show that this type can really represent the type of booleans. Define

$$\texttt{True} = \Lambda\alpha.\lambda x : \alpha.\lambda y : \alpha.x$$
$$\texttt{False} = \Lambda\alpha.\lambda x : \alpha.\lambda y : \alpha.y.$$

Lemma. (3.3) `True` *and* `False` *are the only closed normal terms of type* `Bool`.

Proof. Take M to be a closed normal term of type `Bool`. Lemma 3.2 tells us that M is of the form $\Lambda\alpha.\lambda x : \alpha.\lambda y : \alpha.N$, where $(x : \alpha, y : \alpha) \vdash_\alpha N : \alpha$ and N is normal. It is easily seen that N has to be a variable and so N is either x or y. □

For N_1, N_2, M of respective types σ, σ and `Bool` we define $\mathbf{D}N_1N_2M$ of type σ by

$$\mathbf{D}N_1N_2M = M\{\sigma\}(N_1)(N_2).$$

Now by simple calculation we get

$$\begin{aligned}\mathbf{D}N_1N_2\,\texttt{True} &= (\Lambda\alpha.\lambda x : \alpha.\lambda x : \alpha.x)\{\sigma\}(N_1)(N_2)\\ &\hookrightarrow (\lambda x : \sigma.\lambda y : \sigma.x)(N_1)(N_2)\\ &\hookrightarrow (\lambda y : \sigma.N_1)(N_2)\\ &\hookrightarrow N_1\end{aligned}$$

and

$$\begin{aligned}\mathbf{D}N_1N_2\,\texttt{False} &= (\Lambda\alpha.\lambda x : \alpha.\lambda y : \alpha.x)\{\sigma\}(N_1)(N_2)\\ &\hookrightarrow (\lambda x : \sigma.\lambda y : \sigma.y)(N_1)(N_2)\\ &\hookrightarrow (\lambda y : \sigma.y)(N_2)\\ &\hookrightarrow N_2.\end{aligned}$$

Integer

We define

$$\mathtt{Int} = \prod\alpha.\alpha \to ((\alpha \to \alpha) \to \alpha)$$

and for $n \in \mathbf{N}$ we put

$$\overline{n} = \Lambda\alpha.\lambda x : \alpha.\lambda y : \alpha \to \alpha.\underbrace{y(y \ldots (y}_{n \text{ times}}(x)) \ldots).$$

Lemma. (3.4) *The only closed normal terms of the type* `Int` *are those of the form* $\overline{n}$ *for some* $n \in \mathbf{N}$.

Proof. Similar to the proof of lemma 1.3. Take M to be a closed normal term of type `Int`. By lemma 3.2, M is of the form $\Lambda\alpha.\lambda x : \alpha.\lambda y : \alpha \to \alpha.N$, where $(x : \alpha, y : \alpha \to \alpha) \vdash_\alpha N : \alpha$. N is either a variable (and so x) or is of the form $N_1(N_2)$ (the other cases are immediately ruled out). Now N_1 is a normal term of type $\sigma \to \alpha$. But since N is normal, N_1 cannot be of the form $\lambda z : \sigma.N_1'$ and so N_1 has to be a variable, and hence N_1 is y (and $\sigma = \alpha$). N_2 is a normal term of type α, so it is again either x or y applied to an expression of type α. It follows that N is of the form $y(y \ldots (y(x)) \ldots)$ and so M is of the form $\overline{n}$ for some $n \in \mathbf{N}$. $\square$

We can again define basic functions on `Int`. For example, the successor function $\mathbf{S}$ is defined by

$$\mathbf{S}M = \Lambda\alpha.\lambda x : \alpha.\lambda y : \alpha \to \alpha.(y(M\{\alpha\}(x)(y)))$$

for M of type `Int`. We get

$$\begin{aligned}
\mathbf{S}\overline{n} &= \Lambda\alpha.\lambda x : \alpha.\lambda y : \alpha \to \alpha.(y(\Lambda\alpha.\lambda x : \alpha.\lambda y : \alpha \to \alpha.y(y \ldots (y(x)) \ldots)\{\alpha\}(x)(y))) \\
&\hookrightarrow \Lambda\alpha.\lambda x : \alpha.\lambda y : \alpha \to \alpha.(y(\lambda x : \alpha.\lambda y : \alpha \to \alpha.y(y \ldots (y(x)) \ldots)(x)(y))) \\
&\hookrightarrow \Lambda\alpha.\lambda x : \alpha.\lambda y : \alpha \to \alpha.(y(\lambda y : \alpha \to \alpha.y(y \ldots (y(x)) \ldots)(y))) \\
&\hookrightarrow \Lambda\alpha.\lambda x : \alpha.\lambda y : \alpha \to \alpha.(y(y(y \ldots (y(x)) \ldots))) \\
&= \overline{n+1}.
\end{aligned}$$

Take a closed term f of type $\mathtt{Int} \to \mathtt{Int}$. Then f induces a function $|f|$ from $\mathbf{N}$ to $\mathbf{N}$, defined as follows:

$$|f|(n) = m \qquad \text{iff} \qquad f(\overline{n}) \hookrightarrow^* \overline{m}.$$

Since every term has a unique normal form, this function is well defined and total. Moreover, it is recursive – the algorithm for computing it is simple: write the term $f(\overline{n})$, normalize it (since the terms are strongly normalizable, any normalization strategy will do), observe that the normal form is $\overline{m}$ (by the previous lemma) and put $|f|(n) = m$. The next theorem tells us what is the class of functions representable in second order lambda calculus.

Theorem. (3.5) *The class of functions from* $\mathbf{N}$ *to* $\mathbf{N}$ *which are of the form* $|f|$ *for some* f *is exactly the class of provably total functions in second order Peano arithmetic.*

Proof. For the proof see [**Girard et al. 89**]. $\square$

Product of types

We define

$$\sigma \times \tau = \prod \alpha.(\sigma \to (\tau \to \alpha)) \to \alpha$$

and for M of type σ and N of type τ we put

$$< M, N >= \Lambda\alpha.\lambda x : \sigma \to (\tau \to \alpha).(x(M)(N)).$$

The projections are defined as follows:

$$\pi^1 M = M\{\sigma\}(\lambda x : \sigma.\lambda y : \tau.x)$$
$$\pi^2 M = M\{\tau\}(\lambda x : \sigma.\lambda y : \tau.y).$$

Let us calculate $\pi^1 < M, N >$ and $\pi^2 < M, N >$:

$$\begin{aligned}\pi^1 < M, N > &= (\Lambda\alpha.\lambda x : \sigma \to (\tau \to \alpha).(x(M)(N)))\{\sigma\}(\lambda x : \sigma.\lambda y : \tau.x)\\ &\hookrightarrow (\lambda x : \sigma \to (\tau \to \sigma).(x(M)(N)))(\lambda x : \sigma.\lambda y : \tau.x)\\ &\hookrightarrow (\lambda x : \sigma.\lambda y : \tau.x)(M)(N)\\ &\hookrightarrow (\lambda y : \tau.M)(N)\\ &\hookrightarrow M\end{aligned}$$

$$\begin{aligned}\pi^2 < M, N > &= (\Lambda\alpha.\lambda x : \sigma \to (\tau \to \alpha).(x(M)(N)))\{\tau\}(\lambda x : \sigma.\lambda y : \tau.y)\\ &\hookrightarrow (\lambda x : \sigma \to (\tau \to \tau).(x(M)(N)))(\lambda x : \sigma.\lambda y : \tau.y)\\ &\hookrightarrow (\lambda x : \sigma.\lambda y : \tau.y)(M)(N)\\ &\hookrightarrow (\lambda y : \tau.y)(N)\\ &\hookrightarrow N.\end{aligned}$$

Binary trees

We define

$$\texttt{Bintree} = \prod \alpha.\alpha \to ((\alpha \to (\alpha \to \alpha)) \to \alpha)$$

and for M, N of type `Bintree` we define

$$\begin{aligned}\texttt{nil} &= \Lambda\alpha.\lambda x : \alpha.\lambda y : \alpha \to (\alpha \to \alpha).x\\ \texttt{couple}\, MN &= \Lambda\alpha.\lambda x : \alpha.\lambda y : \alpha \to (\alpha \to \alpha).y(M\{\alpha\}(x)(y))(N\{\alpha\}(x)(y)).\end{aligned}$$

`nil` then represents the tree consisting only of its root and `couple`MN is the tree constructed from the trees M and N by adding a common root.

These are only examples of the considerable expressive power of the system. More examples can be found in [**Girard et al. 89**].

4 Categorical Semantics

In this chapter we give a definition of what constitutes a (categorical) model of a second order lambda calculus (a polymorphic category). Two concepts play a vital role in ideas that led to its definition - that of a cartesian closed category as a model of a simply typed lambda calculus and that of Lawvere's hyperdoctrine. So before we start with the definition of a polymorphic category, we will make a short detour and explain in the following section the latter of these two terms.

4.1 Hyperdoctrines

In [**Lawvere 69**] was introduced the concept of a hyperdoctrine. In it Lawvere for the first time used a method which was later generally considered satisfactory for modelling existentional and universal quantification categorically.

Let us consider a (many-sorted) first order intuitionistic predicate logic based on connectives $\wedge$, $\Rightarrow$ and $\forall$. Such a logic is given by the following:

(1) A many sorted signature Σ specified by
   (i) a set $\mathbf{A}$ of sorts; $\mathbf{A} = \{A, B, \dots\}$.
   (ii) a set $\mathbf{F}$ of operators; $\mathbf{F} = \{f, g, \dots\}$ together with a mapping assigning to each operator its type which is a non-empty type list of sorts - we shall use notation $f : A_1 \dots A_n \to B$.
   (iii) a set $\mathbf{R}$ of relational symbols; $\mathbf{R} = \{R, Q, \dots\}$ together with a mapping assigning to each relational symbol its type - we shall use notation $R \subseteq A_1 \dots A_n$.

(2) For each sort $A \in \mathbf{A}$ we have a countably many variables of that sort. Terms over Σ are defined recursively as follows:
   (i) every variable of sort A is a term of sort A (notation $x : A$.
   (ii) if $f : A_1 \dots A_n \to B$ is an operator and $t_1 : A_1, \dots, t_n : A_n$ are terms, then $f(t_1, \dots, t_n)$ is a term of sort B.

   Context $\mathbf{x}$ is a list of (distinct) variables. A term t is said to be legal in context $\mathbf{x}$ if all variables occuring in t are in $\mathbf{x}$.

(3) Formulas over Σ are given inductively as follows:
   (i) $R(t_1, \dots, t_n)$ is an (atomic) formula for each relational symbol $R \subseteq A_1 \dots A_n$ and terms $t_1 : A_1, \dots, t_n : A_n$.
   (ii) $\top$ is a formula (the truth).
   (iii) if ψ and ϕ are formulas and $x : A$ a variable of sort A, then $\psi \wedge \phi$, $\psi \Rightarrow \phi$, $\forall x : A.\psi$ are formulas.

   A formula ψ is legal in context $\mathbf{x}$ if all free variables of ψ are in $\mathbf{x}$. A sequent is an expression of the form , $\vdash \psi$ where , is a list of formulas. A sequent , $\vdash \psi$ is legal

in a context $\mathbf{x}$ (notation $, \vdash \psi\ (\mathbf{x})$) if $\mathbf{x}$ contains all free variables of $,$ and ψ.

(4) Rules of inference are given below:

$$\text{Weakening}: \quad \frac{, \vdash \psi\ (\mathbf{x})}{, \vdash \psi\ (\mathbf{y})} \quad \text{for } \mathbf{x} \subseteq \mathbf{y}. \qquad \text{Identity}: \quad \frac{}{, , \psi \vdash \psi\ ()}$$

$$\top: \quad \frac{}{, \vdash \top\ ()} \qquad \wedge: \quad \frac{, \vdash \psi\ () \qquad , \vdash \phi\ ()}{, \vdash \psi \wedge \phi\ ()}$$

$$\Rightarrow: \quad \frac{, , \psi \vdash \phi\ ()}{, \vdash \psi \Rightarrow \phi\ ()} \qquad \forall: \quad \frac{, \vdash \psi\ (\mathbf{x}, y)}{, \vdash \forall y\colon A.\psi\ (\mathbf{x})}$$

$$\text{Cut}: \quad \frac{, \vdash \psi\ () \qquad \Delta, \psi \vdash \phi\ ()}{\Delta, , \vdash \phi\ ()}$$

Note that in the $\forall$ rule $y \notin \mathbf{x}$ and y is not a free variable of $,$ or ψ (for the sequent below the line to be well formed).

There are rules of two forms. A collection of sequents is closed under the $\frac{\quad}{\quad}$ rule if whenever it contains the sequents above the line, it also contains the sequent below the line. It is closed under the $\frac{\quad}{\quad}$ rule when it contains the sequents above the thick line if and only if it contains the sequent below the thick line.

Definition. A hyperdoctrine is a tuple (T, P) where

(1) T is a cartesian closed category.

(2) $P\colon T^{\mathrm{OP}} \to \mathbf{Cat}$ is a functor such that

(i) for each object X of T, $P(X)$ is a cartesian closed category.

(ii) for each morphism $f\colon X \to Y$ in T, $P(f)$ preserves the cartesian structure on the nose.

(iii) for each morphism $f\colon X \to Y$ in T, there is given a functor $\prod f\colon P(X) \to P(Y)$ which is the right adjoint to $P(f)\colon P(Y) \to P(X)$.

Now we can interpret the logic in a hyperdoctrine (T, P). This interpretation is given by the following:

(1) A structure M in T for Σ specified by

(i) an object MA of T for each sort A,

(ii) a morphism $Mf\colon MA_1 \times \cdots \times MA_n \to MB$ in T for each operator $f\colon A_1 \ldots A_n \to B$.

(iii) an object MR of $P(MA_1 \times \cdots \times MA_n)$ for each relational symbol $R \subseteq A_1 \ldots A_n$.

(2) Term $t\colon B$ legal in context $\mathbf{x} = (x_1\colon A_1, \ldots, x_n\colon A_n)$ is interpreted by morphism

$$[\![t]\!]_{\mathbf{x}}\colon MA_1 \times \cdots \times MA_n \to MB$$

in T given inductively as follows:

(i) $[\![x_i]\!]_{\mathbf{x}} = \mathbf{snd} \circ \mathbf{fst}^{n-i}$.

(ii) $[\![f(t_1, \ldots, t_m)]\!]_{\mathbf{x}}$ where $f\colon B_1 \ldots B_m \to B$ is the composition

$$MA_1 \times \cdots \times MA_n \xrightarrow{<[\![t_1]\!]_{\mathbf{x}}, \ldots, [\![t_m]\!]_{\mathbf{x}}>} MB_1 \times \cdots \times MB_m \xrightarrow{Mf} MB.$$

(3) Formula ψ legal in context $\mathbf{x} = (x_1 : A_1, \ldots, x_n : A_n)$ is interpreted by object $[\![\psi]\!]_{\mathbf{x}}$ of category $P(MA_1 \times \cdots \times MA_n)$ given inductively as follows:
   (i) $[\![\top]\!]_{\mathbf{x}} = 1_{P(MA_1 \times \cdots \times MA_n)}$ (the terminal object).
   (ii) $[\![R(t_1, \ldots, t_m)]\!]_{\mathbf{x}} = P(< [\![t_1]\!]_{\mathbf{x}}, \ldots, [\![t_m]\!]_{\mathbf{x}} >)(MR)$.
   (iii) $[\![\psi \wedge \phi]\!]_{\mathbf{x}} = [\![\psi]\!]_{\mathbf{x}} \times [\![\phi]\!]_{\mathbf{x}}$ (the categorical product).
   (iv) $[\![\psi \Rightarrow \phi]\!]_{\mathbf{x}} = [\![\psi]\!]_{\mathbf{x}} \to [\![\phi]\!]_{\mathbf{x}}$ (the exponent).
   (v) $[\![\forall y : A.\psi]\!]_{\mathbf{x}} = \prod \mathbf{fst}_{MA_1 \times \cdots \times MA_n, MA}([\![\psi]\!]_{\mathbf{x},y})$.

(4) We say that a sequent $, \vdash \psi$ legal in a context $\mathbf{x}$ is satisfied in the structure M if there is a morphism $p : \prod_{\gamma \in \Gamma} [\![\gamma]\!]_{\mathbf{x}} \to [\![\psi]\!]_{\mathbf{x}}$ in $P(\prod MA)$.

Theorem. (4.1) Soundness *The collection of sequents satisfied in M is closed under the rules of intuitionistic predicate logic.*

Proof. We won't give here a whole proof of this theorem (although it isn't difficult to prove it, the proof is lengthy and tedious). Most of it follows from the fact that a cartesian closed category satisfies the rules of intuitionistic propositional calculus (an analogy of the Soundness theorem for simply typed lambda calculus). What we have here in addition is the $\forall$ rule. We'll show that it is satisfied. First note that when $y \notin \mathbf{x}$ and y is not a free variable of ψ then

$$[\![\psi]\!]_{\mathbf{x},y} = P(\mathbf{fst})[\![\psi]\!]_{\mathbf{x}}.$$

Take a sequent $, \vdash \psi$ legal in context $\mathbf{x}, y$ such that y is not free in ψ or $,$. It is satisfied in a structure M iff there is a morphism

$$p : \prod_{\gamma \in \Gamma} [\![\gamma]\!]_{\mathbf{x},y} \to [\![\psi]\!]_{\mathbf{x},y}.$$

Now

$$\prod_{\gamma \in \Gamma} [\![\gamma]\!]_{\mathbf{x},y} = \prod_{\gamma \in \Gamma} P(\mathbf{fst})[\![\gamma]\!]_{\mathbf{x}} = P(\mathbf{fst})(\prod_{\gamma \in \Gamma} [\![\gamma]\!]_{\mathbf{x}})$$

(since $P(\mathbf{fst})$ preserves the cartesian closed structure on the nose). $\prod \mathbf{fst}$ is a right adjoint to $P(f)$ so there is a morphism

$$p : P(\mathbf{fst})(\prod_{\gamma \in \Gamma} [\![\gamma]\!]_{\mathbf{x}}) \to [\![\psi]\!]_{\mathbf{x},y}$$

iff there is a morphism

$$r : \prod_{\gamma \in \Gamma} [\![\gamma]\!]_{\mathbf{x}} \to [\![\forall y : A.\psi]\!]_{\mathbf{x}}$$

iff M satisfies $, \vdash \forall y : A.\psi$ in context $\mathbf{x}$. So the $\forall$ rule is satisfied. $\square$

Remarks.

(1) The semantics presented above could be extended to a semantics of a logic with $\vee$, $\neg$, $\exists$ connectives and the corresponding rules. We would then need to enrich our definition of a hyperdoctrine by finite sums and left adjoints for all $P(f)$ (the existentional quantification would be interpreted using the left adjoints to $P(\mathbf{fst})$).

(2) We didn't need all the features of a hyperdoctrine (for example we needed only right adjoints to $P(\mathbf{fst})$ and not to all $P(f)$). These features become necessary when we want to model a predicate logic with equality.
(3) Let us mention that the logic presented in this section and its categorical interpretation have also their counterpart among type systems - called the λP system.

4.2 Polymorphic Category

From the beginning the study of the semantics of a second order lambda calculus lagged behind the study of its syntax. Obviously, the main difficulty in modeling a second order lambda calculus lies in the interpretation of universal type $\prod\alpha.\sigma$. We can think of a type as the set of all objects having that type and of a type $\sigma \to \tau$ as the set of functions from σ to τ. What should then the type $\prod\alpha.\sigma$ represent? A typical term of this type is $\Lambda\alpha.M$ where M has type σ. We can apply this term to any type τ and get a term $[\tau/\alpha]M$ of type $[\tau/\alpha]\sigma$. Thus $\Lambda\alpha.M$ is, in fact, a type-indexed family of terms and $\prod\alpha.\sigma$ would be the collection of such families. Therefore our attempt would be to take $\prod\alpha.\sigma$ as the product $\prod_\tau[\tau/\alpha]\sigma$ indexed over all types τ. Putting this in more category-theoretic terms, we would have a small category $\mathbf{Tp}$ with exponentations and products indexed by the set Tp of objects of $\mathbf{Tp}$ and a full and faithful functor $\mathbf{G}$ from $\mathbf{Tp}$ to the category of sets preserving these products and exponentations. In [**Reynolds 84**] is shown by a simple cardinality argument that there are no non-trivial such $\mathbf{Tp}$, $\mathbf{G}$, i.e. that any such $\mathbf{Tp}$ can only contain sets with at most one element. It follows that none of the standard set-theoretic interpretations of the first order typed lambda calculus can be extended to the model of second order lambda calculus. So to be able to find a model of second order lambda calculus, we have to depart from the naive approach of letting λ-terms denote functions and types sets of functions.

In last ten years, there was a considerable progress in the study of semantical problems and a number of models was presented. One solution, quite common, is to follow the way which McCracken used in producing his model based on Scott's model $\mathcal{P}\omega$ [**McCracken 79**], i.e. to start with some suitable model of untyped lambda calculus and interpret types as subsets of this model satisfying some closure properties. To illustrate this method we, in the next section, briefly describe one of these models, the one based on the partial equivalence relation (PER) model of lambda calculus.

In 1986 Girard in [**Girard 86**] presented a different kind of model. It was based on a category of qualitative domains and projection-embedding pairs and in this model types where interpreted, quite pleasingly, as qualitative domains. His ideas can be applied also to other categories of domains what has been done in [**Coquand et al. 87**] for the category of dI-domains, in [**Girard et al. 89**] for coherent domains (coherent spaces) and in [**Coquand et al. 89**] for the category of Scott domains. In the fifth chapter we give a detailed description of one of these models, the one based on the category of Scott domains.

In this kind of models the types with m free variables are interpreted as some functors $(\mathbf{Tp})^m \to \mathbf{Tp}$, where $\mathbf{Tp}$ is a (cartesian closed) category of closed types. In [**Bainbridge et al. 89**] is used yet another approach to the modelling of second order lambda calculus. Types are here interpreted as functors $(\mathbf{Tp}^{\mathrm{OP}})^m \times (\mathbf{Tp})^m \to \mathbf{Tp}$. This idea and its application on

the PER model is briefly discussed in section 4.4.

A general definition of what constitutes a model of a second order lambda calculus was first given by Bruce and Meyer - see [**Bruce et al. 90**]. Later a more categorical definition of a model was proposed by Seely [**Seely 87**] (and similar definition was used by Pitts in [**Pitts 87**]). Definitions given by Seely and Bruce and Meyer are not completely equivalent and now it seems that Seely's definition captures wider range of models, for example the model proposed by Girard is indeed a model in the sence of Seely but it is not a model if we follow verbatim the Bruce and Meyer definition.

The rest of this section is based mainly on paper [**Seely 87**] where Seely gave a definition of a model of higher order polymorphic lambda calculus. The definition we give here is a restriction of Seely's to second order lambda calculus. Some other smaller changes were needed, too (Seely models calculus with product types and sum operator in addition to our implication and universal abstraction). For further references see also [**Pitts 87**] and [**Asperti and Longo 91**].

The starting point is the equivalence between cartesian closed categories and simply typed lambda calculi (with product types). The categorical semantics of second order lambda calculus generalizes this semantics. We need to have some (cartesian closed) category Ω whose objects interpret closed types. Then we have to give some meaning to the context $\Sigma = (\alpha_1, \ldots, \alpha_m)$ and to the types σ legal in this context. Quite natural thing to do is to interpret contexts as products $\Omega^m = \Omega \times \cdots \times \Omega$ and types as morphisms $\sigma : \Omega^m \to \Omega$. Moreover we have to deal with the substitution of a type legal in one context for a type legal in another context. Consider, for example, the type $\vdash_{(\alpha,\beta)} \alpha \to \beta$. Then we can substitute the type $\vdash_\gamma \gamma$ for both α and β to get $\vdash_\gamma \gamma \to \gamma$. So we need a substitution mapping from the types legal in the context (α, β) to the types legal in the context γ (an analogy of the "term" mapping $P(f)$ from the previous section).

When we fix one particular context, the types and the terms legal in this context should form a model of simple typed lambda calculus, i.e. they should form a cartesian closed category, whose objects are the types and morphisms the terms.

The best way how to formalize all this is to use methods of indexed category theory. A model of second order lambda calculus will be a (contravariant) functor $\mathbf{G} : \mathbf{S}^{\mathrm{OP}} \to \mathbf{Cat}$ from some cartesian closed category $\mathbf{S}$ (global) with distinguished object Ω to the category $\mathbf{Cat}$ of small categories such that for every $C \in \mathbf{S}$, the category $\mathbf{G}(C)$ (local) is cartesian closed (intended to interpret types and terms legal in the particular context). Now, as we already mentioned, types legal in a context $\Sigma = (\alpha_1, \ldots, \alpha_m)$ appear both as objects in $\mathbf{G}(\Omega^m)$ and as morphisms $\Omega^m \to \Omega$ in $\mathbf{S}$. So it is natural to require $\mathrm{Obj}\mathbf{G}(C) = \mathrm{Hom}_{\mathbf{S}}(C, \Omega)$. Morphisms $f : C \to D$ in $\mathbf{S}$ should interpret types substitution and so $\mathbf{G}(f) : \mathbf{G}(D) \to \mathbf{G}(C)$ should preserve the cartesian structure (on the nose).

Finally, universal abstraction is interpreted using adjoint functors between local categories in a way presented in the previous section.

We sum up these ideas in the following definition.

Definition. A polymorphic category is a triple $(\mathbf{S}, \mathbf{G}, \Omega)$ where

(1) $\mathbf{S}$ is a cartesian closed category (called global category).
(2) Ω is a distinct object in $\mathbf{S}$.
(3) $\mathbf{G} : \mathbf{S}^{\mathrm{OP}} \to \mathbf{Cat}$ is a functor such that
  (i) for each object C in $\mathbf{S}$, $\mathrm{Obj}\mathbf{G}(C) = \mathrm{Hom}_{\mathbf{S}}(C, \Omega)$ and for each morphism $f : C \to D$, $\mathbf{G}(f) : \mathbf{G}(D) \to \mathbf{G}(C)$ acts on objects as $\mathrm{Hom}_{\mathbf{S}}(f, \Omega)$ (so $\mathbf{G}(f)$ is defined

by composition).

(ii) for each C in $\mathbf{S}$, $\mathbf{G}(C)$ is a cartesian closed category (called local category) and for each $f: C \to D$, $\mathbf{G}(f)$ preserves the cartesian closed structure on the nose.

(iii) for each C in $\mathbf{S}$, there is an adjunction $< \mathbf{G}(\mathbf{fst}_{C,\Omega}), \prod_C, \varphi_C >: \mathbf{G}(C) \to \mathbf{G}(C \times \Omega)$ where $\mathbf{fst}_{C,\Omega}: C \times \Omega \to C$ is the first projection morphism of the binary product in $\mathbf{S}$. Moreover, these adjunctions are natural in C, i.e. for any $f: C \to D$ in $\mathbf{S}$, $\mathbf{G}(f) \circ \prod_D = \prod_C \circ \mathbf{G}(f \times \mathbf{id}_\Omega)$.

Remarks.

(1) We could give a definition of a polymorphic category as a $\mathbf{S}$-indexed category $\mathbf{G}$. Then $< \mathbf{G}(\mathbf{fst}_{\_,\Omega}), \prod, \varphi >$ would form an $\mathbf{S}$-indexed adjunction between $\mathbf{G}$ and $\mathbf{G}^\Omega$ where $\mathbf{G}^\Omega$ is an $\mathbf{S}$-indexed category defined by $\mathbf{G}^\Omega(C) = \mathbf{G}(C \times \Omega)$ and $\mathbf{G}^\Omega(f) = \mathbf{G}(f \times \mathbf{id}_\Omega)$.

(2) Yoneda lemma implies that in a PL category $(\mathbf{S}, \mathbf{G}, \Omega)$ there are morphisms

$$\times_0 : \Omega \times \Omega \to \Omega$$
$$\to_0 : \Omega \times \Omega \to \Omega$$
$$\prod_0 : \Omega^\Omega \to \Omega$$

such that for any $C \in \mathbf{S}$ and for any $f, g \in \mathbf{G}(C) = \mathrm{Hom}_\mathbf{S}(C, \Omega)$

$$f \times_{\mathbf{G}(C)} g = \times_0 \circ < f, g >$$
$$f \to_{\mathbf{G}(C)} g = \to_0 \circ < f, g >$$

and for any $h \in \mathbf{G}(C \times \Omega) = \mathrm{Hom}_\mathbf{S}(C \times \Omega, \Omega)$

$$\prod_C (h) = \prod_0 \circ \lambda h$$

(where $\times_{\mathbf{G}(C)}, \to_{\mathbf{G}(C)}$ denotes the binary product and the exponentation in the local category $\mathbf{G}(C)$ and $\lambda h: C \to \Omega^\Omega$ is the arrow induced by the cartesian structure of $\mathbf{S}$).

We now define the interpretation of the second order lambda calculus in a polymorphic category $(\mathbf{S}, \mathbf{G}, \Omega)$. This interpretation straightforwardly follows the reasoning which led to the definition of a polymorphic category.

Definition. An interpretation of second order lambda calculus in a polymorphic category $(\mathbf{S}, \mathbf{G}, \Omega)$ is defined as follows:

(1) A context $\Sigma = (\alpha_1, \ldots, \alpha_m)$ is interpreted by the object $\Omega^m = ((1_\mathbf{S} \times \Omega) \times \cdots \times \Omega)$ of the global category $\mathbf{S}$. We use notation

$$[\![\Sigma]\!] = \Omega^m$$

(2) A type σ legal in a context $\Sigma = (\alpha_1, \ldots, \alpha_m)$ is interpreted by a morphism

$$[\![\sigma]\!]_\Sigma : [\![\Sigma]\!] \to \Omega$$

in **S** defined inductively as follows:

(i) $[\![\alpha_i]\!]_\Sigma = \mathbf{snd} \circ \mathbf{fst}^{m-i}$

(ii) $[\![\sigma \to \tau]\!]_\Sigma = \times_0 < [\![\sigma]\!]_\Sigma, [\![\tau]\!]_\Sigma >$

(iii) $[\![\prod \alpha.\sigma]\!]_\Sigma = \prod_0 \circ \lambda([\![\sigma]\!]_{\Sigma,\alpha})$

(3) A type assignment $H = (x_1 : \sigma_1, \ldots, x_n : \sigma_n)$ legal in a context $\Sigma = (\alpha_1, \ldots, \alpha_m)$ is interpreted by the product

$$[\![H]\!]_\Sigma = (\cdots(1_{\mathbf{G}([\![\Sigma]\!])} \times [\![\sigma_1]\!]_\Sigma) \times \cdots \times [\![\sigma_n]\!]_\Sigma)$$

in the local category $\mathbf{G}([\![\Sigma]\!])$.

(4) A term $H \vdash_\Sigma M : \sigma$ is interpreted by a morphism

$$[\![M]\!]_{\Sigma;H} : [\![H]\!]_\Sigma \to [\![\sigma]\!]_\Sigma$$

in $G([\![\Sigma]\!])$ defined inductively as follows:

(i) $[\![x_i]\!]_{\Sigma;H} = \mathbf{snd} \circ \mathbf{fst}^{n-i}$

(ii) $[\![\lambda x : \tau.M]\!]_{\Sigma;H} = \lambda([\![M]\!]_{\Sigma;H,x:\tau})$

(iii) $[\![\Lambda\alpha.M]\!]_{\Sigma;H} = \varphi_{[\![\Sigma]\!]}([\![M]\!]_{\Sigma,\alpha;H})$

(iv) $[\![M(N)]\!]_{\Sigma;H} = \mathrm{eval} \circ < [\![M]\!]_{\Sigma;H}, [\![N]\!]_{\Sigma;H} >$

(v) $[\![M\{\tau\}]\!]_{\Sigma;H} = (\mathbf{G}(< \mathbf{id}_{\Omega^m}, [\![\tau]\!]_\Sigma >)(\mathrm{Proj}_{[\![\Sigma]\!]})) \circ [\![M]\!]_{\Sigma;H}$
where $\mathrm{Proj}_{[\![\Sigma]\!]} = \varphi^{-1}_{[\![\Sigma]\!]}(\mathbf{id}_{\Omega^m})$ is the counit of the adjunction

$$< \mathbf{G}(\mathbf{fst}_{[\![\Sigma]\!],\Omega}), \prod_{[\![\Sigma]\!]}, \varphi_{[\![\Sigma]\!]} > : \mathbf{G}([\![\Sigma]\!]) \to \mathbf{G}([\![\Sigma]\!] \times \Omega).$$

Definition. We say that an equation $H \vdash_\Sigma M = N : \sigma$ is satisfied under the given interpretation if $[\![M]\!]_{\Sigma;H} = [\![N]\!]_{\Sigma;H}$.

Theorem. (4.2) (Soundness) *All equational rules for second order lambda calculus are valid under the interpretation given above.*

Proof. This is just a routine check of required equations. (Complete proof of a similar theorem for a model of second order lambda calculus based on Scott domains is given in chapter five). □

4.3 PER Model

In this section we briefly describe model of second order lambda calculus based on the partial equivalence relation model of untyped lambda calculus.

Definition. A partial equivalence relation (per) on a set S is a symmetric and transitive binary relation A on S. For $a, b \in S$ we write aAb to mean that a and b are related by the per A.

Remark. The standard PER model of untyped lambda calculus works with pers on the set **N** of natural numbers. We shall suppose that we have given some enumeration of

partial recursive functions from $\mathbf{N}$ to $\mathbf{N}$ and we shall write f_n for the n-th function in this enumeration. Moreover we suppose that we have given a fixed recursive bijection $<,>: \mathbf{N} \times \mathbf{N} \to \mathbf{N}$.

Definition. PER($\mathbf{N}$) is the following category: its objects are the pers on $\mathbf{N}$. Given $A, B \in \mathrm{PER}(\mathbf{N})$, we say that a morphism from A to B is named by partial recursive function f_n if f_n induces a map of pers from A to B, that is whenever aAb $(a, b \in \mathbf{N})$ then both $f_n(a)$ and $f_n(b)$ are defined and $f_n(a)Bf_n(b)$. Two functions $f_n, f_m : A \to B$ name the same morphism if aAb implies $f_n(a)Bf_m(b)$. Morphisms in PER($\mathbf{N}$) are those named by some functions.

Theorem. (4.3) PER($\mathbf{N}$) *is a cartesian closed category.*

Proof. Product of pers A and B is a per $A \times B$ such that $< a, b > (A \times B) < c, d >$ iff aAc and bBd. Exponent of A and B is the per $A \to B$ such that $a(A \to B)b$ iff f_a and f_b name the same morphism from A to B. Terminal object in PER($\mathbf{N}$) is the per $1_{\mathrm{PER}(\mathbf{N})}$ with one equivalence class. Clearly the given constructions form a cartesian closed structure on $\mathbf{N}$. $\square$

Construction. Now we describe the PER model of second order lambda calculus using the formalization from the previous section. The global category $\mathbf{S}$ has the natural numbers as objects where 1 plays the role of Ω. Morphisms of the global category from m to n are functions $\mathrm{Obj}(\mathrm{PER}(\mathbf{N}))^m \to \mathrm{Obj}(\mathrm{PER}(\mathbf{N}))^n$. Note that in this category 0 is the terminal object and product and exponent of m and n are given by

$$m \times n = m \to n = m + n.$$

The local category $\mathbf{G}(m)$ over $m \in \mathbf{N}$ is then the (cartesian closed) category of functions $\mathrm{Obj}(\mathrm{PER}(\mathbf{N}))^m \to \mathrm{Obj}(\mathrm{PER}(\mathbf{N}))$ with products and exponents computed componentwise. The value of the functor $\prod_m : \mathbf{G}(m+1) \to \mathbf{G}(m)$ on any object $F \in \mathbf{G}(m+1)$ is a function $\prod_m(F) : \mathrm{Obj}(\mathrm{PER}(\mathbf{N}))^m \to \mathrm{Obj}(\mathrm{PER}(\mathbf{N}))$ given by

$$\prod_m(F)(X_1, \ldots, X_m) = \bigcap_{A \in \mathrm{Obj}(\mathrm{PER}(\mathbf{N}))} F(X_1, \ldots, X_m, A).$$

To every term $H \vdash_\Sigma M : \sigma$ where $H = (x_1 : \sigma_1, \ldots, x_n : \sigma_n)$ and $\Sigma = (\alpha_1, \ldots, \alpha_m)$ we can associate a term M^- of untyped lambda calculus simply by erasing the types from M. We proceed by induction on the structure of M:

(1) $(x_i)^- = x_i$
(2) $(\lambda x : \tau . M)^- = \lambda x . M^-$
(3) $(MN)^- = M^- N^-$
(4) $(\Lambda \alpha . M)^- = M^-$
(5) $(M\{\tau\})^- = M^-$

Now to this term M^- we associate a natural number $[\![M^-]\!]_e$ by the following induction:

(1) $[\![x_i]\!]_e$ is (the code of) the partial recursive function $\pi_2 \circ \pi_1^{m-i}$ where π_1, π_2 are the "projection" recursive functions of the pairing bijection $<,> : \mathbf{N} \times \mathbf{N} \to \mathbf{N}$; that is $\pi_1(< a, b >) = a$ and $\pi_2(< a, b >) = b$.
(2) $[\![MN]\!]_e = f_{[\![M]\!]_e}([\![N]\!]_e)$.
(3) $[\![\lambda x . M]\!]_e$ is (the code of) the partial recursive function g such that $f_{g(a)}(b) = f_{[\![M]\!]_e}(< a, b >)$.

A routine check tells us that the $[\![M^-]\!]_e$–th partial recursive function $f_{[\![M^-]\!]_e}$ names a morphism from $[\![\sigma_1]\!]_\Sigma \times \cdots \times [\![\sigma_n]\!]_\Sigma$ to $[\![\sigma]\!]_\Sigma$ in the local category $\mathbf{G}(m)$. So $[\![M^-]\!]_e$ will be the interpretation of the term $H \vdash_\Sigma M : \sigma$. To verify that we have really got the interpretation described in the previous section accounts to checking a few equations.

4.4 Functorial PER Model

In the area of computer science second order lambda calculus was proposed to serve as a syntax for the notion of paramatric polymorphism. Intuitively, a parametric polymorphic function is one that has a uniformly given algorithm for all types.

Example. Consider a function f which takes an argument of a type σ and associates to it an element of a type τ (so that f is of a type $\sigma \to \tau$). For any type α let α-list be the type of lists of elements of the type α and let L be a list of a type σ-list. Consider now the following function **map**: it applies f to the elements of L and then makes a list of results – this is of a type τ-list. Thus **map** has a type $(\sigma \to \tau) \to (\sigma\text{-list} \to \tau\text{-list})$. Note that no specific properties of types σ and τ were used. This map is an example of a function given uniformly for all types - parametric polymorphic function.

In the last few years a great attention was paid to the problem to find a semantic approximation of this notion. A natural starting point is to interpret types as functors $\mathbf{C}^m \to \mathbf{C}$ and terms as natural transformations between them, all defined over some cartesian closed category $\mathbf{C}$ of closed types. The implication operator $\to$ then corresponds to the internal hom functor $\Rightarrow : \mathbf{C}^{\mathrm{OP}} \times \mathbf{C} \to \mathbf{C}$. Unfortunately, this functor is contravariant in the first argument and covariant in the second one and so

$$\alpha \Rightarrow \beta = \Rightarrow \circ <\alpha, \beta> : \mathbf{C}^m \xrightarrow{<\alpha,\beta>} \mathbf{C} \times \mathbf{C} \xrightarrow{\Rightarrow} \mathbf{C}$$

is not a functor (where $\alpha, \beta : \mathbf{C}^m \to \mathbf{C}$).

One solution of this problem is to work with pairs of morphism instead of simple ones which serves to obliterate the difference between covariant and contravariant. This is the way which used Girard in constructing his model and this approach we discuss in the next chapter.

In [**Bainbridge et al. 89**] was proposed another method of solving the problem – not to work with functors $\mathbf{C}^m \to \mathbf{C}$ but rather with functors $(\mathbf{C}^{\mathrm{OP}})^m \times \mathbf{C}^m \to \mathbf{C}$ and in this case the problem described above does not arise. Terms then are not natural but dinatural transformations. These, unfortunately, are not in general composable and so it would not be possible to substitute terms for the free variables in other terms. Neverthless it is possible to find models based on this idea. In [**Bainbridge et al. 89**] is the idea applied to the PER model and worked out in some detail. We give here just a brief description of the model presented there, details and proofs of theorems can be found in the paper.

Definition. A dinatural transformation $\mathbf{u}$ between two functors $F, G : (\mathbf{C}^{\mathrm{OP}})^m \times \mathbf{C}^m \to \mathbf{C}$ is a family of morphisms $\mathbf{u} = \{u_{\mathbf{A}} : F(\mathbf{A}, \mathbf{A}) \to G(\mathbf{A}, \mathbf{A}) | \mathbf{A} \in \mathbf{C}^m\}$ such that, for any vector

of morphisms $\mathbf{f}: \mathbf{A} \to \mathbf{B}$,

$$\begin{array}{ccc} F(\mathbf{A},\mathbf{A}) & \xrightarrow{u_{\mathbf{A}}} & G(\mathbf{A},\mathbf{A}) \\ {\scriptstyle F(\mathbf{f},\mathbf{id}_{\mathbf{A}})}\uparrow & & \downarrow{\scriptstyle G(\mathbf{id}_{\mathbf{A}},\mathbf{f})} \\ F(\mathbf{B},\mathbf{A}) & & G(\mathbf{A},\mathbf{B}) \\ {\scriptstyle F(\mathbf{id}_{\mathbf{B}},\mathbf{f})}\downarrow & & \uparrow{\scriptstyle G(\mathbf{f},\mathbf{id}_{\mathbf{A}})} \\ F(\mathbf{B},\mathbf{B}) & \xrightarrow[u_{\mathbf{B}}]{} & G(\mathbf{B},\mathbf{B}) \end{array}$$

commutes. We use notation $\mathbf{u}: F \dot{\to} G$.

Remark. As we already mentioned, dinatural transformations do not compose in general.

Definition. Given two functors $F, G: \mathbf{C}^m \times \mathbf{C}^m \to \mathbf{C}$. We define $G^F: (\mathbf{C}^{\mathrm{OP}})^m \times \mathbf{C}^m \to \mathbf{C}$ (called twisted exponential) to be the functor

$$(\mathbf{C}^{\mathrm{OP}})^m \times \mathbf{C}^m \xrightarrow{<F^{\mathrm{OP}},G>} \mathbf{C}^{\mathrm{OP}} \times \mathbf{C} \xrightarrow{\Rightarrow} \mathbf{C}.$$

Remark. Recall that $F \times G: (\mathbf{C}^{\mathrm{OP}})^m \times \mathbf{C}^m \to \mathbf{C}$ was defined to be the functor

$$(\mathbf{C}^{\mathrm{OP}})^m \times \mathbf{C}^m \xrightarrow{<F,G>} \mathbf{C} \times \mathbf{C} \xrightarrow{\times} \mathbf{C}.$$

We shall denote by I the subcategory of $\mathrm{PER}(\mathbf{N})$ whose objects are all pers on $\mathbf{N}$ but whose only morphisms are those named by the identity function on $\mathbf{N}$.

Definition. A realizable functor $F: \mathrm{PER}(\mathbf{N}) \to \mathrm{PER}(\mathbf{N})$ is one which takes I to I (setwise) and for which there exists a mapping ϕ from the set of partial recursive functions to itself such that, for any morpfism of pers from A to B named by function f, $F(f)$ is named by $\phi(f)$.

Remark. Realizable functors are closed under products, twisted exponentials and substitution.

Definition. Let. $F, G: (\mathrm{PER}(\mathbf{N})^{\mathrm{OP}})^m \times \mathrm{PER}(\mathbf{N})^m \to \mathrm{PER}(\mathbf{N})$ be realizable functors. A family $\mathbf{u} = \{u_{\mathbf{A}}: F(\mathbf{A},\mathbf{A}) \to G(\mathbf{A},\mathbf{A}) \mid \mathbf{A} \in \mathrm{PER}(\mathbf{N})\}$ (not necessarily dinatural) is called a realizable family if there is a single partial recursive function φ such that each morphism $u_{\mathbf{A}}$ is denoted by φ.

Theorem. (4.4) *Realizable dinatural transformations compose.*

Corollary. (4.5) *For each $m \in \mathbf{N}$, the realizable functors $(\mathrm{PER}(\mathbf{N})^{\mathrm{OP}})^m \times \mathrm{PER}(\mathbf{N})^m \to \mathrm{PER}(\mathbf{N})$ and the realizable dinatural transformations between them form a cartesian closed category.*

Definition. Given a realizable functor $G: (\mathrm{PER}(\mathbf{N})^{\mathrm{OP}})^m \times \mathrm{PER}(\mathbf{N})^m \to \mathrm{PER}(\mathbf{N})$, the realizable end of G (denoted $\int_{\mathbf{A}} G(\mathbf{A},\mathbf{A})$ where $\mathbf{A} \in \mathrm{PER}(\mathbf{N})^m$) is a per $E \in \mathrm{PER}(\mathbf{N})$

together with realizable dinatural transformation $\omega\colon \mathbf{K}_E \stackrel{\cdot\cdot}{\to} G$ universal for all such realizable dinatural transformations (where $\mathbf{K}_E\colon (\mathrm{PER}(\mathbf{N})^{\mathrm{OP}})^m \times \mathrm{PER}(\mathbf{N})^m \to \mathrm{PER}(\mathbf{N})$ is the constant functor with value E). That is, for any $D \in \mathrm{PER}(\mathbf{N})$ and for any realizable dinatural transformation $\beta\colon \mathbf{K}_D \stackrel{\cdot\cdot}{\to} G$ there is a unique realizable dinatural transformation $h\colon \mathbf{K}_d \stackrel{\cdot\cdot}{\to} \mathbf{K}_E$ such that

$$\beta_{\mathbf{A}} = \omega_{\mathbf{A}} \circ h_{\mathbf{A}}$$

for any $\mathbf{A} \in \mathrm{PER}(\mathbf{N})^m$.

Remarks.

(1) The given definition of a realizable end of a realizable functor is a special case of more general definition of an end of a functor. Its special case for covariant functor is just the definition of a limit of a functor. The realizable end of a functor will provide an interpretation of universal types.

(2) For $G\colon (\mathrm{PER}(\mathbf{N})^{\mathrm{OP}})^{m+1} \times \mathrm{PER}(\mathbf{N})^{m+1} \to \mathrm{PER}(\mathbf{N})$, we write $\int_Q G(-,Q,-,Q)$, where $Q \in \mathrm{PER}(\mathbf{N})$, for the functor $(\mathrm{PER}(\mathbf{N})^{\mathrm{OP}})^m \times \mathrm{PER}(\mathbf{N})^m \to \mathrm{PER}(\mathbf{N})$ whose value on $(\mathbf{A},\mathbf{B})$, where $\mathbf{A},\mathbf{B} \in \mathrm{PER}(\mathbf{N})^m$, is the end $\int_Q(\mathbf{A},Q,\mathbf{B},Q)$ of the functor

$$G(\mathbf{A},-,\mathbf{B},-)\colon \mathrm{PER}(\mathbf{N})^{\mathrm{OP}} \times \mathrm{PER}(\mathbf{N}) \to \mathrm{PER}(\mathbf{N}).$$

Theorem. (4.6) *For any realizable functor* $G\colon (\mathrm{PER}(\mathbf{N})^{\mathrm{OP}})^m \times \mathrm{PER}(\mathbf{N})^m \to \mathrm{PER}(\mathbf{N})$ *the realizable end* $\int_{\mathbf{A}} G(\mathbf{A},\mathbf{A})$ *exists.*

Now we give a description of an interpretation of second order lambda calculus in the terms of realizable functors and realizable dinatural transformations.

Definition. An interpretation $[\![\sigma]\!]_\Sigma$ of a type σ legal in the context $\Sigma = (\alpha_1,\ldots,\alpha_m)$ is a realizable functor $(\mathrm{PER}(\mathbf{N})^{\mathrm{OP}})^m \times \mathrm{PER}(\mathbf{N})^m \to \mathrm{PER}(\mathbf{N})$ given inductively as follows:

(1) $[\![\alpha_i]\!]_\Sigma$ is the projection functor onto ith covariant argument, i.e. $[\![\alpha_i]\!]_\Sigma(\mathbf{A},\mathbf{B}) = B_i$.

(2) $[\![\sigma \to \tau]\!]_\Sigma = [\![\tau]\!]_\Sigma^{[\![\sigma]\!]_\Sigma}$ (twisted exponential).

(3) $[\![\prod \alpha.\tau]\!]_\Sigma$ is the realizable functor given by

$$[\![\prod \alpha.\tau]\!]_\Sigma(\mathbf{A},\mathbf{B}) = \int_Q [\![\tau]\!]_{\Sigma,\alpha}(\mathbf{A},Q,\mathbf{B},Q)$$

where Q is any per.

Terms are interpreted in the same way as in the previous section. It is easy to verify that $[\![M^-]\!]_e$ names a realizable dinatural transformation

$$[\![M^-]\!]_e\colon ([\![\sigma_1]\!]_\Sigma \times \cdots \times [\![\sigma_n]\!]_\Sigma) \stackrel{\cdot\cdot}{\to} [\![\sigma]\!]_\Sigma\colon (\mathrm{PER}(\mathbf{N})^{\mathrm{OP}})^m \times \mathrm{PER}(\mathbf{N})^m \to \mathrm{PER}(\mathbf{N}).$$

5 Scott Domains Model

In 1986 J. Y. Girard in his paper [**Girard 86**] used a new method for producing a model of second order lambda calculus where types were interpreted as qualitative domains. Later ideas presented in this paper were applied to other categories of domains, namely to the category of dI-domains in [**Coquand et al. 87**], the category of complete algebraic lattices in [**Coquand and Ehrhard 87**], the category of coherence spaces [**Girard et al. 89**] and the category of Scott domains [**Coquand et al. 89**]. In this chapter we describe in detail the probably most interesting model of these, the one based on Scott domains.

5.1 Scott Domains and Categories

In this section we review some basic definitions and results concerning Scott domains as well as a few more notions of category theory. Proofs can be found in any paper concerning Scott domains.

Definition. Let $(D, \leq)$ be a partially ordered set. We say that

(1) $(D, \leq)$ is directed if it is nonempty and, for any $i, j \in D$, there is $k \in D$ such that $i \leq k$ and $j \leq k$.
(2) $(D, \leq)$ is complete (and we say that $(D, \leq)$ is a cpo) if it has a least element $\perp$ and every directed subset $M \subseteq D$ has a least upper bound $\bigvee M$.
(3) a point $x \in D$ is finite if for every directed $M \subseteq D$ such that $x \leq \bigvee M$ there is $y \in M$ such that $x \leq y$. Let $\mathbf{B}_D$ denote the set of finite elements of D.
(4) $(D, \leq)$ is algebraic if, for every $x \in D$, the set $M = \{x_0 \in \mathbf{B}_D \mid x_0 \leq x\}$ is directed and $x = \bigvee M$.
(5) $(D, \leq)$ is bounded complete if every bounded subset of D has a least upper bound.

Definition. A Scott domain is a bounded complete algebraic cpo.

Remark. The least upper bounds of finite sets of finite elements are finite, when they exist.

Definition. Let $(D, \leq_D)$, $(E, \leq_E)$ be partially ordered sets. A function $f : D \to E$ is monotonic if it preserves the order, that is $x \leq_D y$ implies $f(x) \leq_E f(y)$.
A monotonic function $f : D \to E$ is continuous if $f(\bigvee M) = \bigvee f(M)$ for any directed $M \subseteq D$.

Remark. Scott domains and continuous functions between them form an important cartesian closed category $\mathbf{D}$. Product of two domains D and E is the domain $D \times_{\mathbf{D}} E$ of pairs of elements (d, e) where $d \in D$ and $e \in E$ ordered coordinatewise with the obvious projections $\mathbf{fst}_{D,E}$ and $\mathbf{snd}_{D,E}$. Their exponent $D \to_{\mathbf{D}} E$ is a domain consisting of the continuous

functions from D to E ordered pointwise, i.e.

$$f \leq g \quad \text{iff} \quad \forall d \in D, f(d) \leq g(d).$$

Definition. A pair of continuous functions (f, g) where $f\colon D \to E$ and $g\colon E \to D$ between two cpos D and E is called embedding-projection pair if, for all $d \in D$, $g \circ f(d) = d$ and, for all $e \in E$, $f \circ g(e) \leq e$.

Remark. If $h = (f, g)$ is an embedding-projection pair then f is called the embedding (notation $h^{\mathrm{L}} = f$) and g the projection ($h^{\mathrm{R}} = g$). As embedding-projection pair forms a special case of adjoint functors (between two categories which are partially ordered sets) the embedding determines its accompanying projection uniquely and vice versa. Scott domains and embedding-projection pairs form a cartesian closed category $\mathbf{D}^{\mathrm{EP}}$. The composition of two embedding-projection pairs is defined in the obvious way: for $h = h^{\mathrm{L}}, h^{\mathrm{R}}) \in \mathbf{D}^{\mathrm{EP}}(D, E)$ and $k = (k^{\mathrm{L}}, k^{\mathrm{R}}) \in \mathbf{D}^{\mathrm{EP}}(E, F)$ we put

$$k \circ h = (k^{\mathrm{L}} \circ h^{\mathrm{L}}, h^{\mathrm{R}} \circ k^{\mathrm{R}}) \in \mathbf{D}^{\mathrm{EP}}(D, F).$$

The identity morphism of a domain D is the pair $(\mathbf{id}_D, \mathbf{id}_D)$. The cartesian structure of $\mathbf{D}^{\mathrm{EP}}$ is then given by the same constructions as in the category $\mathbf{D}$.

Definition. A colimit $< \rho_i \in \mathbf{D}^{\mathrm{EP}}(D_i, D) >_{i \in I}$ in $\mathbf{D}^{\mathrm{EP}}$ is directed if the indexing poset I is directed. A category is directed complete if it has colimits of all directed families.

Remark. So a cpo is directed complete when regarded as a category.

Theorem. (5.1) *The category $\mathbf{D}^{\mathrm{EP}}$ is directed complete. A cone $< \rho_i \in \mathbf{D}^{\mathrm{EP}}(D_i, D) >_{i \in I}$ is a directed colimit iff $\{\rho_i^{\mathrm{L}} \circ \rho_i^{\mathrm{R}} \mid i \in I\}$ is directed in $D \to_{\mathbf{D}} D$ and*

$$\mathbf{id}_D = \bigvee \{\rho_i^{\mathrm{L}} \circ \rho_i^{\mathrm{R}} \mid i \in I\}.$$

Theorem. (5.2) *Let D be a domain. Then*

$$\{f^{\mathrm{L}} \circ f^{\mathrm{R}} \mid f \in \mathbf{D}^{\mathrm{EP}}(X, D) \text{ for some finite domain } X\}$$

is a directed subset of finite elements in $D \to_{\mathbf{D}} D$ and

$$\mathbf{id}_D = \bigvee \{f^{\mathrm{L}} \circ f^{\mathrm{R}} \mid f \in \mathbf{D}^{\mathrm{EP}}(X, D) \text{ for some finite domain } X\}.$$

Remark. From theorem (5.2) it follows that a domain is a colimit of the finite domains which embed into it.

Lemma. (5.3) *Let $f_0 \in \mathbf{D}^{\mathrm{EP}}(X_0, D)$ and $f_1 \in \mathbf{D}^{\mathrm{EP}}(X_1, D)$, where X_0, X_1 are finite domains. Then there is a finite domain X and $g \in \mathbf{D}^{\mathrm{EP}}(X, D)$ so that $g_0 = (g^{\mathrm{R}} \circ f_0^{\mathrm{L}}, f_0^{\mathrm{R}} \circ g^{\mathrm{L}}) \in \mathbf{D}^{\mathrm{EP}}(X_0, X)$ and $g_1 = (g^{\mathrm{R}} \circ f_1^{\mathrm{L}}, f_1^{\mathrm{R}} \circ g^{\mathrm{L}}) \in \mathbf{D}^{\mathrm{EP}}(X_1, X)$ with $f_0 = g \circ g_0$ and $f_1 = g \circ g_1$.*

Lemma. (5.4) *Suppose* $< \rho_i \in \mathbf{D}^{\mathrm{EP}}(D_i, D) >_{i \in I}$ *is a directed colimit in* $\mathbf{D}^{\mathrm{EP}}$. *If* X *is a finite domain and* $f \in \mathbf{D}^{\mathrm{EP}}(X, D)$ *then there is some* $i \in I$ *and* $h \in \mathbf{D}^{\mathrm{EP}}(X, D_i)$ *such that* $f = \rho_i \circ h$.

Definition. A functor $F : C \to C'$ between directed complete categories is continuous if it preserves directed colimits.

Remarks.

(1) We define the following two continuous functors $\times, \to : \mathbf{D}^{\mathrm{EP}} \times \mathbf{D}^{\mathrm{EP}} \to \mathbf{D}^{\mathrm{EP}}$:
The product operator $\times$ is defined by

$$\begin{aligned} \times(A, B) &= A \times B \\ \times(f, g) &= f \times g : A \times A' \to B \times B' \end{aligned}$$

where $A, B \in \mathbf{D}^{\mathrm{EP}}$, $f \in \mathbf{D}^{\mathrm{EP}}(A, B)$ and $g \in \mathbf{D}^{\mathrm{EP}}(A', B')$.
The function operator $\to$ is defined by

$$\begin{aligned} \to(A, B) &= A \to B \\ \to(f, g) &= f \to g \in \mathbf{D}^{\mathrm{EP}}(A \to A', B \to B') \end{aligned}$$

where

$$(f \to g)^{\mathrm{L}}(h) = g^{\mathrm{L}} \circ h \circ f^{\mathrm{R}}$$

for $h \in \mathbf{D}(A, A')$ and

$$(f \to g)^{\mathrm{R}}(h') = g^{\mathrm{R}} \circ h' \circ f^{\mathrm{L}}$$

for $h' \in \mathbf{D}(B, B')$.

(2) In order to cope with the presence of free type variables it is convenient to define generalizations of the product and function space functors. Given $F, G : \mathbf{C} \to \mathbf{D}^{\mathrm{EP}}$ we define

$$\begin{aligned} F \# G &= \times \circ (F \times G) \circ \Delta : \mathbf{C} \xrightarrow{\Delta} \mathbf{C} \times \mathbf{C} \xrightarrow{F \times G} \mathbf{D}^{\mathrm{EP}} \times \mathbf{D}^{\mathrm{EP}} \xrightarrow{\times} \mathbf{D}^{\mathrm{EP}} \\ F \Rightarrow G &= \to \circ (F \times G) \circ \Delta : \mathbf{C} \xrightarrow{\Delta} \mathbf{C} \times \mathbf{C} \xrightarrow{F \times G} \mathbf{D}^{\mathrm{EP}} \times \mathbf{D}^{\mathrm{EP}} \xrightarrow{\to} \mathbf{D}^{\mathrm{EP}} \end{aligned}$$

where Δ is the diagonal functor. We can also define a multiary version of the $\#$ by taking $\#()$ to be the functor into the trivial domain $1_{\mathbf{D}^{\mathrm{EP}}}$ and setting

$$\#(F_1, \ldots, F_{n+1}) = \#(F_1, \ldots, F_n) \# F_{n+1}.$$

(3) Given functors $F_1, \ldots, F_n$ and $1 \leq i \leq n$ we define

$$\mathbf{p}_X^{i,n} : F_1(X) \times \cdots \times F_n(X) \to F_i(X)$$

to be the ith projection of the product $F_1(X) \times \cdots \times F_n(X)$.

(4) In order to simplify notation we shall write $F^{\mathrm{L}}(f)$ (resp. $F^{\mathrm{R}}(f)$) for $(F(f))^{\mathrm{L}}$ (resp. $(F(f))^{\mathrm{R}}$). To reduce the number of parentheses we shall also assume that association is to the left so that expressions such as fxy or $f(x)(y)$ represent an expression $(f(x))(y)$.

5.2 Continuous Sections

Now we come to the central part of the model. In this section we describe a construction which allows us to interpret universal abstraction. In [**Coquand et al. 89**] this is done in more general way working with Grothendieck fibrations and their continuous sections. Since it is not essential for understanding the model, we omit the general definition of Grothendieck fibration and restrict ourselves to its special form in the category $\mathbf{D}^{\mathrm{EP}}$.

Definition. Let $F\colon \mathbf{C} \to \mathbf{D}^{\mathrm{EP}}$ be a continuous functor from a directed complete category $\mathbf{C}$. We define category $\prod F$ of continuous sections of (the Grothendieck cofibration of) F as follows:

(1) Objects of $\prod F$ are families $< t_X >_{X \in \mathbf{C}}$, where $t_X \in F(X)$, satisfying the following conditions:

(i) If $f \in \mathbf{C}(X, Y)$ then

$$F^{\mathrm{L}}(f) t_X \leq t_Y \qquad \text{(monotonicity)}.$$

(ii) If $< \rho_i \in \mathbf{C}(X_i, X) >_{i \in I}$ is a directed colimit in $\mathbf{C}$ then

$$t_X = \bigvee_{i \in I} F^{\mathrm{L}}(\rho_i) t_{X_i} \qquad \text{(continuity)}.$$

(2) There is at most one morphism between two continuous sections t and t' – we write this morphism as $t \leq t'$ and define

$$t \leq t' \qquad \text{iff} \qquad \forall X \in \mathbf{C}. t_X \leq t'_X.$$

Remark. Recalling Theorem (5.1) we can, for a functor $F\colon \mathbf{D}^{\mathrm{EP}} \to \mathbf{D}^{\mathrm{EP}}$, rewrite the condition (ii) as follows: If for a cone $< \rho_i \in \mathbf{D}^{\mathrm{EP}}(X_i, X) >_{i \in I}$ we have $\{\rho_i^{\mathrm{L}} \circ \rho_i^{\mathrm{R}} \mid i \in I\}$ is directed in $X \to X$ and $\bigvee_{i \in I} \rho_i^{\mathrm{L}} \circ \rho_i^{\mathrm{R}} = \mathbf{id}_X$ then $t_X = \bigvee_{i \in I} F^{\mathrm{L}}(\rho_i) t_{X_i}$. We shall later use this form of the condition (ii).

Since our aim is to interpret closed types as domains and types with one free variable as continuous functors $F\colon \mathbf{D}^{\mathrm{EP}} \to \mathbf{D}^{\mathrm{EP}}$ we need, in particular, $\prod F$ to be one. Unfortunately, it is not quite as its objects are not sets. But we can put the objects of $\prod F$ in 1-1 correspondence with the elements of a suitable set. Take $\mathbf{S}$ to be some countable subcategory of $\mathbf{D}^{\mathrm{EP}}$ equivalent to the full subcategory of all finite domains (with embedding-projection pairs as morphisms). Any continuous section is determined by its restriction to the domains of $\mathbf{S}$. So in this sence $\prod F$ is isomorphic to a partially ordered set and in fact, as we shall see, to a domain.

Theorem. (5.5) *Let $F\colon \mathbf{D}^{\mathrm{EP}} \to \mathbf{D}^{\mathrm{EP}}$ be a continuous functor. Then the category $\prod F$ is isomorphic to a Scott domain (regarded as a category).*

Proof. Take $\prod_{\mathbf{S}} F$ to be the partial order consisting of monotonic families $< t_X >_{X \in \mathbf{S}}$, i.e. families satisfying condition (i). Clearly, $\prod_{\mathbf{S}} F$ is a set because $\mathbf{S}$ is. Now we show that $\prod F$ and $\prod_{\mathbf{S}} F$ are isomorphic (as categories) and, later, that $\prod_{\mathbf{S}} F$ is a domain.

Any continuous section $t \in \prod F$ determines, by restriction, an element $\mathrm{res}\, t \in \prod_{\mathbf{S}} F$. Conversely, any $t \in \prod_{\mathbf{S}} F$ can be extended to a continuous section $\mathrm{ext}\, t \in \prod F$ by taking

$$(\mathrm{ext}\ t)_D = \bigvee \{F^{\mathrm{L}}(f)t_X \mid X \in \mathbf{S} \ \ \& \ \ f \in \mathbf{D}^{\mathrm{EP}}(X, D)\}$$

for any domain D. We must check that this is well defined.

First of all we check that the set $\bigvee\{F^{\mathrm{L}}(f)t_X \mid X \in \mathbf{S} \ \ \& \ \ f \in \mathbf{D}^{\mathrm{EP}}(X, D)\}$ is directed (and so the least upper bound exists). For any two elements of this set $y_0 = F^{\mathrm{L}}(f_0)t_{X_0}$ and $y_1 = F^{\mathrm{L}}(f_1)t_{X_1}$ arising from morphisms $f_0 \in \mathbf{D}^{\mathrm{EP}}(X_0, D)$ and $f_1 \in \mathbf{D}^{\mathrm{EP}}(X_1, D)$, X_0, X_1 are finite, there is, by Lemma (5.3), a finite domain X and morphisms $g \in \mathbf{D}^{\mathrm{EP}}(X, D)$, $g_0 \in \mathbf{D}^{\mathrm{EP}}(X_0, X)$, $g_1 \in \mathbf{D}^{\mathrm{EP}}(X_1, X)$ such that $f_0 = g \circ g_0$ and $f_1 = g \circ g_1$. t is monotonic and so $y_0 = F^{\mathrm{L}}(g \circ g_0)t_{X_0} \leq F^{\mathrm{L}}(g)t_X$. Similarly $y_1 \leq F^{\mathrm{L}}(g)t_X$ and hence the set $\bigvee\{F^{\mathrm{L}}(f)t_X \mid X \in \mathbf{S} \ \ \& \ \ f \in \mathbf{D}^{\mathrm{EP}}(X, D)\}$ is directed and the definition above defines a family. It remains to show that this family is monotonic and continuous. For any $g \in \mathbf{D}^{\mathrm{EP}}(D, E)$ we get

$$\begin{aligned}
F^{\mathrm{L}}(g)(\mathrm{ext}\ t)_D &= F^{\mathrm{L}}(g) \bigvee \{F^{\mathrm{L}}(f)t_X \mid X \in \mathbf{S} \ \ \& \ \ f \in \mathbf{D}^{\mathrm{EP}}(X, D)\} \\
&= \bigvee \{F^{\mathrm{L}}(g) \circ F^{\mathrm{L}}(f)t_X \mid X \in \mathbf{S} \ \ \& \ \ f \in \mathbf{D}^{\mathrm{EP}}(X, D)\} \\
&= \bigvee \{F^{\mathrm{L}}(g \circ f)t_X \mid X \in \mathbf{S} \ \ \& \ \ f \in \mathbf{D}^{\mathrm{EP}}(X, D)\} \\
&\leq \bigvee \{F^{\mathrm{L}}(h)t_X \mid X \in \mathbf{S} \ \ \& \ \ h \in \mathbf{D}^{\mathrm{EP}}(X, E)\} \\
&= (\mathrm{ext}\ t)_E.
\end{aligned}$$

This shows the monotonicity. Take a directed colimit $< \rho_i \in \mathbf{D}^{\mathrm{EP}}(D_i, D) >_{i \in I}$. We need to show that

$$(\mathrm{ext}\ t)_D = \bigvee_{i \in I} \{F^{\mathrm{L}}(\rho_i)(\mathrm{ext}\ t)_{D_i} \mid i \in I\}.$$

The set is directed because ext t is monotonic. From monotonicity we also get

$$(\mathrm{ext}\ t)_D \geq \bigvee_{i \in I} \{F^{\mathrm{L}}(\rho_i)(\mathrm{ext}\ t)_{D_i} \mid i \in I\}.$$

$(\mathrm{ext}\ t)_D = \bigvee\{F^{\mathrm{L}}(f)t_X \mid X \in \mathbf{S} \ \ \& \ \ f \in \mathbf{D}^{\mathrm{EP}}(X, D)\}$. Take an element $F^{\mathrm{L}}(F)t_X$ of the set on the right hand side. By Lemma (5.5), there is $i \in I$ and $f \in \mathbf{D}^{\mathrm{EP}}(X, D)$ such that $f = \rho_i \circ h$. Now we have

$$\begin{aligned}
F^{\mathrm{L}}(f)t_X &= F^{\mathrm{L}}(\rho_i \circ h)t_X \\
&= f^{\mathrm{L}}(\rho_i)(F^{\mathrm{L}}(h)t_X) \\
&\leq F^{\mathrm{L}}(\rho_i)(\mathrm{ext}\ t)_{D_i}.
\end{aligned}$$

It follows that the other inequality $(\mathrm{ext}\ t)_D \leq \bigvee\{F^{\mathrm{L}}(\rho_i)(\mathrm{ext}\ t)_{D_i} \mid i \in I\}$ holds and hence ext t is continuous.

It is easy to see that the two operations $\mathrm{res} : \prod F \to \prod_{\mathbf{S}} F$ and $\mathrm{ext} : \prod_{\mathbf{S}} F \to \prod F$ preserve the order relation (and so are functors when the domains regarded as categories).

Take $t \in \prod_{\mathbf{S}} F$. Then, for $Y \in \mathbf{S}$, we have $t_Y \leq (\mathrm{ext}\ t)_Y$ – to see this , take in the definition of ext t f to be the identity on Y. From the monotonicity of T we get

$$(\mathrm{res\ ext}\ t)_Y = \bigvee\{F^{\mathrm{L}}(f)t_X \mid X \in \mathbf{S} \ \ \& \ \ f \in \mathbf{D}^{\mathrm{EP}}(X, Y)\} \leq t_Y$$

and hence res ext $t = t$. Conversely, for $t \in \prod F$, we have $(\text{res } t)_X = t_X$ for any $X \in \mathbf{S}$. Then, from the definition of ext we obtain

$$(\text{ext res } t)_D = \bigvee\{F^{\mathrm{L}}(f)t_X \mid X \in \mathbf{S} \ \ \& \ \ f \in \mathbf{D}^{\mathrm{EP}}(X, D)\}$$

for any domain D. But since t is continuous and D is the directed colimit of finite embeddings (see the remark following Theorem (5.2)), we also have

$$t_D = \bigvee\{F^{\mathrm{L}}(f)t_X \mid X \in \mathbf{S} \ \ \& \ \ f \in \mathbf{D}^{\mathrm{EP}}(X, D)\}.$$

Hence ext res $t = t$. So res and ext form an isomorphism.

Now we proceed to show that $\prod_{\mathbf{S}} F$ is a domain. The least element is the family $<\bot_X>_{X\in\mathbf{S}}$. Suppose $\{t^i \mid i \in I\}$ is a directed subset of $\prod_{\mathbf{S}} F$. Take t to be the family defined by

$$t_X = \bigvee_{i\in I} t^i_X,$$

for $X \in \mathbf{S}$. The least upper bound on the right exists because the set $\{t^i_X \mid i \in I\}$ is directed in $F(X)$. t is monotonic because, for $f \in \mathbf{D}^{\mathrm{EP}}(X, Y)$,

$$F^{\mathrm{L}}(f)t_X = F^{\mathrm{L}}(f)(\bigvee_{i\in I} t^i_X) = \bigvee_{i\in I} F^{\mathrm{L}}(f)t^i_X \leq \bigvee_{i\in I} t^i_Y,$$

where we used the fact that $F^{\mathrm{L}}(f)$ is continuous. Hence t is the least upper bound of the set $\{t^i \mid i \in I\}$ and so $\prod_{\mathbf{S}} F$ is complete. To show that it is bounded complete we proceed similarly. Suppose $\{t^i \mid i \in I\}$ is bounded by s, i.e. $t^i \leq s$ for $i \in I$ and define t by taking

$$t_X = \bigvee_{i\in I} t^i_X.$$

Now the least upper bound exists because $\{t^i_X \mid i \in I\}$ is bounded in $F(X)$. It is again monotonic –

$$F^{\mathrm{L}}(f)t_X = F^{\mathrm{L}}(f)(\bigvee_{i\in I} t^i_X) = \bigvee_{i\in I} F^{\mathrm{L}}(f)t^i_X \leq \bigvee_{i\in I} t^i_Y,$$

using the fact that embeddings preserve all existing upper bounds.

It remains to show that $\prod_{\mathbf{S}} F$ is algebraic. Suppose there is $t \in \prod_{\mathbf{S}} F$ such that $t_X = e \in F(X)$ is finite for some $X \in \mathbf{S}$. Define

$$[S, e]_Y = \bigvee\{F^{\mathrm{L}}(f)e \mid f \in \mathbf{D}^{\mathrm{EP}}(X, Y)\}$$

for $Y \in \mathbf{S}$. This is well defined (t_Y is a bound for the set on the right) that it is monotonic and does not depend on the choice of t. Consider a family

$$s = [X_1, e_1] \vee \cdots \vee [X_n, e_n].$$

We show that s is a finite element of $\prod_{\mathbf{S}} F$. Suppose $s \leq \bigvee M$ where M is directed subset of $\prod_{\mathbf{S}} F$. Then, for any $1 \leq i \leq n$, we get

$$e_{X_i} \leq s_{X_i} \leq (\bigvee M)_{X_i} = \bigvee_{m \in M} m_{X_i}.$$

As e_i is finite, $e_i \leq m^i_{X_i}$ for some $m^i \in M$. But then $[X_i, e_i] \leq m^i$. Since M is directed, there is $m \in M$ which dominates each m^i for $1 \leq i \leq n$ and so $s \leq m$. Hence s is finite.

Any $t \in \prod_{\mathbf{S}} F$ is easily seen to be the least upper bound of the directed set

$$\{[X_1, e_1] \vee \cdots \vee [X_n, e_n] \mid e_1 \leq t_{X_i} \ \& \ \cdots \ \& \ e_n \leq t_{X_n}\}$$

where the least upper bounds $[X_1, e_1] \vee \cdots \vee [X_n, e_n]$ exist because they are bounded above. Since in a domain least upper bouns of finite sets of finite elements are finite (when they exist), it follows that any finite element of $\prod_{\mathbf{S}} F$ must be of the form $[X_1, e_1] \vee \cdots \vee [X_n, e_n]$. Hence $\prod_{\mathbf{S}} F$ is algebraic and so a domain. $\square$

Remarks.

(1) In the following we shall treat $\prod F$ as a domain. It would be possible to replace everywhere $\prod F$ by $\prod_{\mathbf{S}} F$ provided above but this would only make the text even more complicated.

(2) We will need to use $\prod$ operator with parameters. If $F \colon (\mathbf{D}^{\mathrm{EP}})^{m+1} \to \mathbf{D}^{\mathrm{EP}}$ is a continuous functor, we shall write $\prod^m F \colon (\mathbf{D}^{\mathrm{EP}})^m \to \mathbf{D}^{\mathrm{EP}}$ for the continuous functor defined as follows:

$$(\textstyle\prod^m F)(A) = \prod(F(A, -))$$

for $A \in (\mathbf{D}^{\mathrm{EP}})^m$. Given $f \in (\mathbf{D}^{\mathrm{EP}})^m(A, B)$, we put

$$(\textstyle\prod^m F)(f) \in \mathbf{D}^{\mathrm{EP}}((\prod^m F)(A), (\prod^m F)(B))$$

by taking

$$(\textstyle\prod^m F)^{\mathrm{L}}(f)(s)_Z = F^{\mathrm{L}}(f, \mathbf{id}_Z)(s_Z)$$
$$(\textstyle\prod^m F)^{\mathrm{R}}(f)(t)_Z = F^{\mathrm{R}}(f, \mathbf{id}_Z)(t_Z)$$

for each section $s \in (\prod^m F)(A)$ and $t \in ((\prod^m F)(B)$. Checking that this is well defined is a routine. Note that $\prod^0 F$ for $F \colon \mathbf{D}^{\mathrm{EP}} \to \mathbf{D}^{\mathrm{EP}}$ is $\prod F$.

Now we introduce notation and results needed to provide a semantics of second order lambda calculus which is done in the next section.

Lemma. (5.6) *Suppose $F_1, \ldots, F_n \colon (\mathbf{D}^{\mathrm{EP}})^m \to \mathbf{D}^{\mathrm{EP}}$ are continuous functors. Then $\mathbf{p}^{i,n}$ is a continuous section of the functor $\#(F_1, \ldots, F_n) \Rightarrow F_i$.*

Proof. For definition of $\mathbf{p}^{i,n}$ see the end of section 5.1. Suppose $f \in (\mathbf{D}^{\mathrm{EP}})^m(X, Y)$ and $(x_1, \ldots, x_n) \in F_1(X) \times \cdots \times F_n(X)$. Recall that

$$(\#(F_1, \ldots, F_n) \Rightarrow F_i)(X) = \#(F_1(X) \times \cdots \times F_n(X)) \to_{\mathbf{D}^{\mathrm{EP}}} F_i(X)$$

and so really $\mathbf{p}^{i,n}_X \in (\#(F_1,\ldots,F_n)\Rightarrow F_i)(X)$. Then

$$\begin{aligned}(\#(F_1,\ldots,F_n)\Rightarrow F_i)^{\mathrm{R}}(f)(\mathbf{p}^{i,n}_Y)(x_1,\ldots,x_n)&\\ &= (F_i^{\mathrm{R}}(f)\circ \mathbf{p}^{i,n}_Y \circ \#(F_1,\ldots,F_n)^{\mathrm{L}}(f))(x_1,\ldots,x_n)\\ &= F_i^{\mathrm{R}}(f)(F_i^{\mathrm{L}}(f)(x_i))\\ &= x_i\\ &= \mathbf{p}^{i,n}_X(x_1,\ldots,x_n)\end{aligned}$$

and we get

$$\begin{aligned}(\#(F_1,\ldots,F_n)\Rightarrow F_i)^{\mathrm{L}}(f)(\mathbf{p}^{i,n}_X)(y_1,\ldots,y_n)&\\ &= (\#(F_1,\ldots,F_n)\Rightarrow F_i)^{\mathrm{L}}(f)(\#(F_1,\ldots,F_n)\Rightarrow F_i)^{\mathrm{R}}(f)(\mathbf{p}^{i,n}_Y)(y_1,\ldots,y_n)\\ &\leq \mathbf{p}^{i,n}_Y(y_1,\ldots,y_n).\end{aligned}$$

Hence $\mathbf{p}^{i,n}$ is monotonic and it is easy to see that it is continuous: Let $f_j \in (\mathbf{D}^{\mathrm{EP}})^m(X_j, X)$, for $j \in J$ such that $\{f_j^{\mathrm{L}} \circ f_j^{\mathrm{R}}\}$ form a directed collection such that $\bigvee_j f_j^{\mathrm{L}} \circ f_j^{\mathrm{R}} = \mathbf{id}_X$. Then

$$\begin{aligned}\bigvee_{j\in J}(\#(F_1,\ldots,F_n)\Rightarrow F_i)^{\mathrm{L}}(f_j)(\mathbf{p}^{i,n}_{X_j})(x_1,\ldots,x_n)&\\ &= \bigvee_{j\in J}(F_i^{\mathrm{L}}(f_j)\circ \mathbf{p}^{i,n}_{X_j} \circ \#(F_1,\ldots,F_n)^{\mathrm{R}}(f_j))(x_1,\ldots,x_n)\\ &= \bigvee_{j\in J}(F_i^{\mathrm{L}}(f_j)(F_i^{\mathrm{R}}(f_j))(x_i)\\ &= x_i\\ &= \mathbf{p}^{i,n}_X(x_1,\ldots,x_n). \quad \square\end{aligned}$$

Suppose $P, F, G\colon (\mathbf{D}^{\mathrm{EP}})^m \to \mathbf{D}^{\mathrm{EP}}$ are continuous functors. Suppose also that s is a continuous section of the functor $P\Rightarrow(F\Rightarrow G)\colon (\mathbf{D}^{\mathrm{EP}})^m \to \mathbf{D}^{\mathrm{EP}}$ and t is a continuous section of the functor $P\Rightarrow F\colon (\mathbf{D}^{\mathrm{EP}})^m \to \mathbf{D}^{\mathrm{EP}}$. Then define a family $\mathbf{apply}(s,t)$ by

$$\mathbf{apply}(s,t)_X(x) = (s_X(x))(t_X(x)),$$

where $X \in (\mathbf{D}^{\mathrm{EP}})^m$ and $x \in P(X)$.

Lemma. (5.7) $\mathbf{apply}(s,t)$ *is a continuous section of the functor* $P\Rightarrow G\colon (\mathbf{D}^{\mathrm{EP}})^m \to \mathbf{D}^{\mathrm{EP}}$.

Proof. To show that $\mathbf{apply}(s,t)$ is monotonic, suppose $f \in (\mathbf{D}^{\mathrm{EP}})^m(X,Y)$. Then for

$y \in P(Y)$ we have

$$\begin{aligned}
&(P{\Rightarrow}G)^{\mathrm{L}}(f)(\mathbf{apply}(s,t)_X)(x)\\
&\quad = (G^{\mathrm{L}}(f) \circ (\mathbf{apply}(s,t)_X) \circ P^{\mathrm{R}}(f))(x)\\
&\quad = G^{\mathrm{L}}(f)((\mathbf{apply}(s,t)_X)(P^{\mathrm{R}}(f)(x)))\\
&\quad = G^{\mathrm{L}}(f)((s_X(P^{\mathrm{R}}(f)(x)))(t_X(P^{\mathrm{R}}(f)(x))))\\
&\quad \le G^{\mathrm{L}}(f)(((F{\Rightarrow}G)^{\mathrm{R}}(f)(s_Y(x)))(F^{\mathrm{R}}(f)(t_Y(x))))\\
&\quad = G^{\mathrm{L}}(f)((G^{\mathrm{R}}(f) \circ (s_Y(x)) \circ F^{\mathrm{L}}(f))(F^{\mathrm{R}}(f)(t_Y(x))))\\
&\quad \le (s_Y(x))(t_Y(x))\\
&\quad = \mathbf{apply}(s,t)_Y(x).
\end{aligned}$$

To show that $\mathbf{apply}(s,t)$ is continuous , suppose we have $f_i \in (\mathbf{D}^{\mathrm{EP}})^m(X_i, X)$ for $i \in I$ such that $\{f_i^{\mathrm{L}} \circ f_i^{\mathrm{R}}\}$ is directed in $X \to_{\mathbf{D}^{\mathrm{EP}}} X$ and $\bigvee_i f_i^{\mathrm{L}} \circ f_i^{\mathrm{R}} = \mathbf{id}_X$. Then for $x \in P(X)$

$$\begin{aligned}
&\bigvee_i (P{\Rightarrow}G)^{\mathrm{L}}(f_i)(\mathbf{apply}(s,t)_{X_i})(x)\\
&\quad = \bigvee_i G^{\mathrm{L}}(f_i)((s_{X_i}(P^{\mathrm{R}}(f_i)(x)))(t_{X_i}(P^{\mathrm{R}}(f_i)(x))))\\
&\quad = \bigvee_i G^{\mathrm{L}}(f_i)(((F{\Rightarrow}G)^{\mathrm{R}}(f_i)((P{\Rightarrow}(F{\Rightarrow}G))^{\mathrm{L}}(f_i)(s_{X_i})(x)))\\
&\qquad (F^{\mathrm{R}}(f_i)((P{\Rightarrow}F)^{\mathrm{L}}(f_i)(t_{X_i})(x))))\\
&\quad = \bigvee_i G^{\mathrm{L}}(f_i)(((F{\Rightarrow}G)^{\mathrm{R}}(f_i)(s_X(x)))(F^{\mathrm{R}}(f_i)(t_X(x))))\\
&\quad = \bigvee_i ((G^{\mathrm{L}}(f_i) \circ G^{\mathrm{R}}(f_i))(s_X(x)))((F^{\mathrm{L}}(f_i) \circ F^{\mathrm{R}}(f_i))(t_X(x))\\
&\quad = \mathbf{apply}(s,t)_X(x). \quad \square
\end{aligned}$$

Suppose $P, G\colon (\mathbf{D}^{\mathrm{EP}})^m \to \mathbf{D}^{\mathrm{EP}}$, $F\colon (\mathbf{D}^{\mathrm{EP}})^{m+1} \to \mathbf{D}^{\mathrm{EP}}$ are continuous functors and t is a continuous section of the functor $P{\Rightarrow}\prod^m F\colon (\mathbf{D}^{\mathrm{EP}})^m \to \mathbf{D}^{\mathrm{EP}}$. We define a family $\mathbf{Apply}(t,G)$ by

$$\mathbf{Apply}(t,G)_X(x) = t_X(x)_{G(X)},$$

where $x \in P(X)$.

Lemma. (5.8) $\mathbf{Apply}(t,G)$ *is a continuous section of the functor* $P{\Rightarrow}(F \circ < \mathbf{Id}_{(\mathbf{D}^{\mathrm{EP}})^m}, G >)\colon (\mathbf{D}^{\mathrm{EP}})^m \to \mathbf{D}^{\mathrm{EP}}$.

Proof. In a similar way as in the proofs of the previous two lemmas we show that $\mathbf{Apply}(t,G)$ is indeed monotonic and continuous. $\square$

Let $P, F, G\colon (\mathbf{D}^{\mathrm{EP}})^m \to \mathbf{D}^{\mathrm{EP}}$ be continuous functors. Suppose t is a continuous section of the functor $(P\#F){\Rightarrow}G\colon (\mathbf{D}^{\mathrm{EP}})^m \to \mathbf{D}^{\mathrm{EP}}$. Take a family $\mathbf{curry}(t)$ defined by

$$\mathbf{curry}(t)_X(x)(y) = t_X(x,y),$$

where $x \in P(X)$ and $y \in F(X)$.

Lemma. (5.9) $\mathbf{curry}(t)$ *is a continuous section of the functor* $P \Rightarrow (F \Rightarrow G) : (\mathbf{D}^{\mathrm{EP}})^m \to \mathbf{D}^{\mathrm{EP}}$.

Proof. In a similar way as the proof of Lemmma 5.7. $\square$

Suppose $P : (\mathbf{D}^{\mathrm{EP}})^m \to \mathbf{D}^{\mathrm{EP}}$ and $F : (\mathbf{D}^{\mathrm{EP}})^{m+1} \to \mathbf{D}^{\mathrm{EP}}$ are continuous functors. Let t be a continuous section of the functor $(P \circ \mathbf{Fst}_{(\mathbf{D}^{\mathrm{EP}})^m, \mathbf{D}^{\mathrm{EP}}}) \Rightarrow F : (\mathbf{D}^{\mathrm{EP}})^{m+1} \to \mathbf{D}^{\mathrm{EP}}$. Let $X \in (\mathbf{D}^{\mathrm{EP}})^m$ and $x \in P(X)$. We define $\mathbf{Curry}(t)_X(x)$ to be the continuous section of $F(X, -) : \mathbf{D}^{\mathrm{EP}} \to \mathbf{D}^{\mathrm{EP}}$ given by the equation

$$\mathbf{Curry}(t)_X(x)_Z = t_{(X,Z)}(x),$$

where $Z \in \mathbf{D}^{\mathrm{EP}}$.

Lemma. (5.10) $\mathbf{Curry}(t)$ *is a continuous section of the functor* $P \Rightarrow \prod^m F : (\mathbf{D}^{\mathrm{EP}})^m \to \mathbf{D}^{\mathrm{EP}}$.

Proof. Similar as the proof of Lemmma 5.7. $\square$

Suppose $P, F, G : (\mathbf{D}^{\mathrm{EP}})^m \to \mathbf{D}^{\mathrm{EP}}$ are continuous functors and suppose we have given s to be a continuous section of $(P \# F) \Rightarrow G$ and t to be a continuous section of $P \Rightarrow F$. Then we define a continuous section $[t]s$ of $P \Rightarrow F$ by

$$([t]s)_X(x) = \mathbf{apply}(\mathbf{curry}(s), t)(x) = s_X(x, t_X(x)),$$

where $x \in P(X)$.

Lemma. (5.11)

(1) *Take* t, s, t', s' *to be continuous sections of functors* $P \Rightarrow F, ((P \# F) \# F) \Rightarrow G$, $((P \# F) \# F) \Rightarrow G$ *and* $((P \# F) \Rightarrow F$, *respectively. If* $t'_X(p, b) = t_X(p)$ *and* $s'_X(p, b, a) = s_X(p, a, b)$ *for every* X, p, a *and* b, *then* $\mathbf{curry}([t'], s') = [t](\mathbf{curry}(s))$.

(2) *If* $t'_{(X,Y)} = t_X$, *then* $\mathbf{Curry}([t']s) = [t](\mathbf{Curry}(s))$ *(where* t, s, t' *are continuous sections of appropriate functors).*

(3) $\mathbf{apply}([t]r, [t]s) = [t](\mathbf{apply}(r, s))$.

(4) $\mathbf{Apply}([t]s, G) = [t](\mathbf{Apply}(s, G))$.

Proof.

(1)

$$\begin{aligned}
\mathbf{curry}([t']s')_X(p)(b) &= ([t']s')_X(p, b) \\
&= s'_X(p, b, t'_X(p, b)) \\
&= s_X(p, t_X(p), b) \\
&= \mathbf{curry}(x)_X(p, t_X(p))(b) \\
&= ([t](\mathbf{curry}(s)))_X(p)(b).
\end{aligned}$$

(2)

$$\begin{aligned}\mathbf{Curry}([t']s)_X(x)_Y &= s_{(X,Y)}(x, t'_{(X,Y)}(x)) \\ &= s_{(X,Y)}(x, t_X(x))_Y \\ &= [t](\mathbf{Curry}(s))_X(x)_Y .\end{aligned}$$

(3)

$$\begin{aligned}\mathbf{apply}([t]r, [t]s)_X(x) &= ([t]r_X(x))([t]s_X(x)) \\ &= (r_X(x, t_X(x)))(s_X(x, t_X(x))) \\ &= \mathbf{apply}(r, s)_X(x, t_X(x)) \\ &= [t](\mathbf{apply}(r, s))_X(x).\end{aligned}$$

(4)

$$\begin{aligned}\mathbf{Apply}([t]s, G)_X(x) &= ([t]s)_X(x)_{G(X)} \\ &= s_X(x, t_X(x))_{G(X)} \\ &= \mathbf{Apply}(s, G)_X(x, t_X(x)) \\ &= [t](\mathbf{Apply}(s, G))_X(x). \quad \square\end{aligned}$$

Suppose that $P, K : (\mathbf{D}^{\mathrm{EP}})^m \to \mathbf{D}^{\mathrm{EP}}$ and $F : (\mathbf{D}^{\mathrm{EP}})^{m+1} \to \mathbf{D}^{\mathrm{EP}}$ are continuous functors and that t is a continuous section of $(P \circ \mathbf{Fst}_{(\mathbf{D}^{\mathrm{EP}})^m, \mathbf{D}^{\mathrm{EP}}}) \Rightarrow F)$. We define a continuous section $[K]t$ of $P \Rightarrow (F \circ < \mathbf{Id}_{(\mathbf{D}^{\mathrm{EP}})^m}, G >)$ by

$$([K]t)_X(x) = \mathbf{Apply}(\mathbf{Curry}(t), K)_X(x) = t_{(X, G(X))}(x).$$

Lemma. (5.12)

(1) $\mathbf{curry}([K]t) = [K](\mathbf{curry}(t))$.
(2) *If* $t'_{(X,Z,Y)} = t_{(X,Y,Z)}$ *for each* X, Y *and* Z, *then* $\mathbf{Curry}([K \circ \mathbf{Fst}]t') = [K](\mathbf{Curry}(t))$.
(3) $\mathbf{apply}([K]s, [K]t) = [K](\mathbf{apply}(s, t))$.
(4) $\mathbf{Apply}([K]t, H \circ < \mathbf{Id}, K >) = [K](\mathbf{Apply}(t, H))$.

Proof.

(1)

$$\begin{aligned}\mathbf{curry}([K]t)_X(x)(y) &= ([K]t)_X(x, y) \\ &= t_{(X, K(X))}(x, y) \\ &= \mathbf{curry}(t)_{(X, K(X))}(x)(y) \\ &= [K](\mathbf{curry}(t))_X(x)(y).\end{aligned}$$

(2)

$$\begin{aligned}\mathbf{Curry}([K\circ\mathbf{Fst}]t)_X(x)_Z &= \mathbf{Curry}([K\circ\mathbf{Fst}]t')_{(X,Z)}\\ &= t'_{(X,Z,G(X))}(x)\\ &= t_{(X,G(X),Z)}(x)_Z\\ &= \mathbf{Curry}(t)_{(X,G(X))}(x)_Z\\ &= [K](\mathbf{Curry}(t))_X(x)_Z.\end{aligned}$$

(3)

$$\begin{aligned}\mathbf{apply}([K]s,[K]t)_X(x) &= (([K]s)_X(x))(([K]t)_X(x))\\ &= (s_{(X,K(X))}(x))(t_{(X,K(X))}(x))\\ &= (\mathbf{apply}(s,t))_{(X,K(X))}(x)\\ &= [K](\mathbf{apply}(s,t))_X(x).\end{aligned}$$

(4)

$$\begin{aligned}\mathbf{Apply}([K]t,H\circ<\mathbf{Id},K>) &= ([K]t)_X(x)_{H(X,K(X))}\\ &= t_{(X,K(X))}(x)_{H(X,K(X))}\\ &= \mathbf{Apply}(t,H)_{(X,K(X))}(x)\\ &= [K](\mathbf{Apply}(t,H))_X(x).\quad\square\end{aligned}$$

5.3 Semantics

Now we finally describe the semantics for second order lambda calculus in this model. As we already mentioned, types are interpreted as functors from $(\mathbf{D}^{\mathrm{EP}})^m$ to $\mathbf{D}^{\mathrm{EP}}$.

Definition.

(1) An interpretation $[\![\sigma]\!]_\Sigma$ of a type σ legal in a context $\Sigma=(\alpha_1,\ldots,\alpha_m)$ is a continuous functor $(\mathbf{D}^{\mathrm{EP}})^m\to\mathbf{D}^{\mathrm{EP}}$ defined inductively as follows:
  (i) $[\![\alpha_i]\!]_\Sigma=\mathbf{P}^{i,m}_{(\mathbf{D}^{\mathrm{EP}})^m}$
  (ii) $[\![\sigma\to\tau]\!]_\Sigma=[\![\sigma]\!]_\Sigma\Rightarrow[\![\tau]\!]_\Sigma$
  (iii) $[\![\prod\alpha.\sigma]\!]_\Sigma=\prod^m([\![\sigma]\!]_\Sigma)$.

(2) A type assignment $H=(x_1:\sigma_1,\ldots,x_n:\sigma_n)$ legal in a context $\Sigma=(\alpha_1,\ldots,\alpha_m)$ is interpreted by the functor $[\![H]\!]_\Sigma=\#([\![\sigma_1]\!]_\Sigma,\ldots,[\![\sigma_n]\!]_\Sigma)$.

(3) An interpretation $[\![M]\!]_{\Sigma;H}$ of the term $H\vdash_\Sigma M:\sigma$ is a continuous section of the functor $[\![H]\!]_\Sigma\Rightarrow[\![\sigma]\!]_\Sigma\colon(\mathbf{D}^{\mathrm{EP}})^m\to\mathbf{D}^{\mathrm{EP}}$. It is defined inductively as follows:
  (i) $[\![x_i]\!]_{\Sigma;H}=\mathbf{p}^{i,n}$
  (ii) $[\![\lambda x:\tau.M]\!]_{\Sigma;H}=\mathbf{curry}([\![M]\!]_{\Sigma;H,x:\tau})$
  (iii) $[\![\Lambda\alpha.M]\!]_{\Sigma;H}=\mathbf{Curry}([\![M]\!]_{\Sigma,\alpha;H})$
  (iv) $[\![M(N)]\!]_{\Sigma;H}=\mathbf{apply}([\![N]\!]_{\Sigma;H},[\![M]\!]_{\Sigma;H})$
  (v) $[\![M\{\tau\}]\!]_{\Sigma;H}=\mathbf{Apply}([\![M]\!]_{\Sigma;H},[\![\tau]\!]_\Sigma)$.

Remark. A detailed inspection shows that these definitions really make sence, the only problem is the case (iv). This is dealt with in the following lemma.

Lemma. (5.13) *If α does not appear free in the type σ, then*

$$[\![\sigma]\!]_{\Sigma,\alpha} = [\![\sigma]\!]_{\Sigma} \circ \mathbf{Fst}_{(\mathbf{D}^{\mathrm{EP}})^m,\mathbf{D}^{\mathrm{EP}}}.$$

Proof. By straightforward induction on σ.

$$[\![\alpha_i]\!]_{\Sigma,\alpha} = \mathbf{P}^{i,m+1} = [\![\alpha_i]\!]_{\Sigma} \circ \mathbf{Fst}_{(\mathbf{D}^{\mathrm{EP}})^m,\mathbf{D}^{\mathrm{EP}}}.$$

$$\begin{aligned}
[\![\sigma \to \tau]\!]_{\Sigma,\alpha} &= [\![\sigma]\!]_{\Sigma,\alpha} \Rightarrow [\![\tau]\!]_{\Sigma,\alpha} \\
&= ([\![\sigma]\!]_{\Sigma} \circ \mathbf{Fst}_{(\mathbf{D}^{\mathrm{EP}})^m,\mathbf{D}^{\mathrm{EP}}}) \Rightarrow ([\![\tau]\!]_{\Sigma} \circ \mathbf{Fst}_{(\mathbf{D}^{\mathrm{EP}})^m,\mathbf{D}^{\mathrm{EP}}}) \\
&= ([\![\sigma]\!]_{\Sigma} \Rightarrow [\![\tau]\!]_{\Sigma}) \circ \mathbf{Fst}_{(\mathbf{D}^{\mathrm{EP}})^m,\mathbf{D}^{\mathrm{EP}}} \\
&= [\![\sigma \to \tau]\!]_{\Sigma} \circ \mathbf{Fst}_{(\mathbf{D}^{\mathrm{EP}})^m,\mathbf{D}^{\mathrm{EP}}}
\end{aligned}$$

$$\begin{aligned}
[\![\textstyle\prod \beta.\sigma]\!]_{\Sigma,\alpha}(X,Y) &= \textstyle\prod^{m+1} [\![\sigma]\!]_{\Sigma,\alpha,\beta}(X,Y) \\
&= \textstyle\prod ([\![\sigma]\!]_{\Sigma,\alpha,\beta}(X,Y,-)) \\
&= \textstyle\prod ([\![\sigma]\!]_{\Sigma,\beta,\alpha}(X,-,Y) \\
&= \textstyle\prod ([\![\sigma]\!]_{\Sigma,\beta} \circ \mathbf{Fst}_{(\mathbf{D}^{\mathrm{EP}})^{m+1},\mathbf{D}^{\mathrm{EP}}}(X,-,Y)) \\
&= \textstyle\prod ([\![\sigma]\!]_{\Sigma,\beta}(X,-)) \\
&= \textstyle\prod^{m} ([\![\sigma]\!]_{\Sigma,\beta})(X) \\
&= \textstyle\prod ([\![\sigma]\!]_{\Sigma,\beta}) \circ \mathbf{Fst}_{(\mathbf{D}^{\mathrm{EP}})^m,\mathbf{D}^{\mathrm{EP}}}(X,Y) \\
&= [\![\sigma]\!]_{\Sigma,\beta} \circ \mathbf{Fst}_{(\mathbf{D}^{\mathrm{EP}})^m,\mathbf{D}^{\mathrm{EP}}}(X,Y),
\end{aligned}$$

where the fourth line follows, for $\beta \neq \alpha$, from the induction hypothesis and, for $\beta = \alpha$, from the fact that

$$[\![\sigma]\!]_{\Sigma,\alpha,\alpha} = [\![\sigma]\!]_{\Sigma,\alpha} \circ \mathbf{Fst}_{(\mathbf{D}^{\mathrm{EP}})^m,\mathbf{D}^{\mathrm{EP}}}. \quad \square$$

Example. The interpretation of the type $\prod \alpha.\alpha \to \alpha$ is

$$\begin{aligned}
[\![\textstyle\prod \alpha.\alpha \to \alpha]\!] &= \textstyle\prod^{0} ([\![\alpha \to \alpha]\!]_{\alpha}) \\
&= \textstyle\prod ([\![\alpha]\!]_{\alpha} \Rightarrow [\![\alpha]\!]_{\alpha}) \\
&= \textstyle\prod (\mathbf{P}^{1,1} \Rightarrow \mathbf{P}^{1,1}).
\end{aligned}$$

The polymorphic identity function is interpreted by the following continuous section of $\prod(\mathbf{P}^{1,1} \Rightarrow \mathbf{P}^{1,1})$:

$$\begin{aligned}
[\![\Lambda\alpha.\lambda x : \alpha.x]\!] &= \mathbf{Curry}([\![\lambda x : \alpha.x]\!]_{\alpha}) \\
&= \mathbf{Curry}(\mathbf{curry}([\![x]\!]_{\alpha;x:\alpha})) \\
&= \mathbf{Curry}(\mathbf{curry}(\mathbf{p}^{1,1})).
\end{aligned}$$

Definition. We say that equation $H \vdash_\Sigma M = N : \sigma$ is satisfied under the given semantics if $[\![M]\!]_{\Sigma;H} = [\![N]\!]_{\Sigma;H}$.

We now proceed to prove Soundness theorem. First we need some lemmas.

Lemma. (5.14) *Given permutations* $\{1,\dots,n\} = \{i_1,\dots,i_n\}$ *and* $\{1,\dots,m\} = \{j_1,\dots,j_m\}$. *Then*

$$[\![M]\!]_{\alpha_1,\dots,\alpha_m;x_1:\sigma_1,\dots,x_n:\sigma_n(X_1,\dots,X_m)}(p_1,\dots,p_n) = \\ = [\![M]\!]_{\alpha_{j_1},\dots,\alpha_{j_m};x_{i_1}:\sigma_{i_1},\dots,x_{i_n}:\sigma_{i_n}(X_{j_1},\dots,X_{j_m})}(p_{i_1},\dots,p_{i_n}).$$

Proof. By structural induction on M. □

Lemma. (5.15) *Suppose* $H \vdash_\Sigma M_1 : \sigma_1$ *and* $H, x:\sigma_1 \vdash_\Sigma M_2 : \sigma_2$. *Then*

$$\mathbf{apply}(\mathbf{curry}([\![M_2]\!]_{\Sigma;H,x:\sigma_1}), [\![M_1]\!]_{\Sigma;H}) = [\![[M_1/x]M_2]\!]_{\Sigma;H}.$$

Proof. Let $r = [\![[M_1/x]M_2]\!]_{\Sigma;H}$, $s = [\![M_2]\!]_{\Sigma;H,x:\sigma_1}$ and $t = [\![M_1]\!]_{\Sigma;H}$. We want to show that $r = [t]s$. We shall do it by structural induction on M_2.

(1) $M_2 \equiv x$. Then $r = t$ and $[t]s = [t](\mathbf{p}^{n+1,n+1}) = t$, so $r = [t]s$.

(2) $M_2 \equiv x_i$. Then $r = [\![x_i]\!]_{\Sigma;H} = \mathbf{p}^{i,n} = [t](\mathbf{p}^{i,n+1}) = t$.

(3) $M_2 \equiv \lambda y:\sigma.M$. Suppose that $\sigma_2 = \sigma \to \tau$ so that $H, y:\sigma \vdash_\Sigma M : \tau$. Then

$$\begin{aligned} r &= [\![\lambda y:\sigma.[M_1/x]M]\!]_{\Sigma;H} \\ &= \mathbf{curry}([\![[M_1/x]M]\!]_{\Sigma;H,y:\sigma}) \\ &= \mathbf{curry}([[\![M_1]\!]_{\Sigma;H,y:\sigma}][\![M]\!]_{\Sigma;H,y:\sigma,x:\sigma_1}) \qquad \text{(hyp)} \\ &= [t](\mathbf{curry}([\![M]\!]_{\Sigma;H,x:\sigma_1,y:\sigma})) \qquad \text{(Lemmas 5.11.1 and 5.14)} \\ &= [t]s. \end{aligned}$$

(4) $M_2 \equiv \Lambda\alpha.M$. Suppose that $\sigma_2 = \prod\alpha.\sigma$ so that $H \vdash_{\Sigma,\alpha} M : \sigma$. Then

$$\begin{aligned} r &= [\![\Lambda\alpha.[M_1/x]M]\!]_{\Sigma;H} \\ &= \mathbf{Curry}([\![[M_1/x]M]\!]_{\Sigma,\alpha;H}) \\ &= \mathbf{Curry}([[\![M_1]\!]_{\Sigma,\alpha;H}][\![M]\!]_{\Sigma,\alpha;H,x:\sigma_1}) \qquad \text{(hyp)} \\ &= [t](\mathbf{Curry}([\![M]\!]_{\Sigma,\alpha;H,x:\sigma_1})) \qquad \text{(Lemmas 5.11.2 and 5.14)} \\ &= [t]s. \end{aligned}$$

(5) $M_2 \equiv M(N)$. Suppose that $H \vdash_\Sigma M : \sigma \to \sigma_2$ and $H \vdash_\Sigma N : \sigma$. Then

$$\begin{aligned} r &= [\![([M_1/x]M)([M_1/x]N)]\!]_{\Sigma;H} \\ &= \mathbf{apply}([\![[M_1/x]M]\!]_{\Sigma;H}, [\![[M_1/x]N]\!]_{\Sigma;H}) \\ &= \mathbf{apply}([t][\![M]\!]_{\Sigma;H,x:\sigma_1}, [t][\![N]\!]_{\Sigma;H,x:\sigma_1}) \qquad \text{(hyp)} \\ &= [t](\mathbf{apply}([\![M]\!]_{\Sigma;H,x:\sigma_1}, [\![N]\!]_{\Sigma;H,x:\sigma_1})) \qquad \text{(Lemma 5.11.3)} \\ &= [t]s. \end{aligned}$$

(6) $M_2 \equiv M\{\sigma\}$. Suppose $H \vdash_\Sigma M : \tau$. Then

$$\begin{aligned} r &= [\![([M_1/x]M)\{\sigma\}]\!]_{\Sigma;H} \\ &= \mathbf{Apply}([\![[M_1/x]M]\!]_{\Sigma;H}, [\![\sigma]\!]_\Sigma) \\ &= \mathbf{Apply}([t][\![M]\!]_{\Sigma;H,x:\sigma_1}, [\![\sigma]\!]_\Sigma) \qquad \text{(hyp)} \\ &= [t](\mathbf{Apply}([\![M]\!]_{\Sigma;H,x:\sigma_1}, [\![\sigma]\!]_\Sigma) \qquad \text{(Lemma 5.11.4)} \\ &= [t]s. \quad \square \end{aligned}$$

Lemma. (5.16) $[\![[\sigma_2/\alpha]\sigma_1]\!]_\Sigma = [\![\sigma_1]\!]_{\Sigma,\alpha} \circ < \mathbf{Id}_{(\mathbf{D}^{\mathrm{EP}})^m}, [\![\sigma_2]\!]_\Sigma >$.

Proof. By structural induction on σ_1. $\square$

Lemma. (5.17) *Suppose $H \vdash_{\Sigma,\alpha} M : \sigma_1$ and α does not appear free in type of variable in H. Then*

$$\mathbf{Apply}(\mathbf{Curry}([\![M]\!]_{\Sigma,\alpha;H})[\![\sigma_2]\!]_\Sigma) = [\![[\sigma_2/\alpha]M]\!]_{\Sigma;H}.$$

Proof. Let $s = [\![[\sigma_2/\alpha]M]\!]_{\Sigma;H}$, $t = [\![M]\!]_{\Sigma,\alpha;H}$ and $K = [\![\sigma_2]\!]_\Sigma$. By structural induction on M we show that $s = [K]t$.

(1) $M \equiv x_i$. Then $[\sigma/\alpha]\sigma_1 = \sigma_1$, and so $s = [\![x_i]\!]_{\Sigma;H} = \mathbf{p}^{i,n} = [K](\mathbf{p}^{i,n+1}) = [K]t$.

(2) $M \equiv \lambda y\colon \sigma.N$. Suppose that $\sigma_1 = \sigma \to \tau$ so that $H \vdash_{\Sigma,\alpha} N : \tau$. Then

$$\begin{aligned} s &= [\![\lambda y\colon [\sigma_2/\alpha]\sigma.[\sigma_2/\alpha]N]\!]_{\Sigma;H} \\ &= \mathbf{curry}([\![[\sigma_2/\alpha]N]\!]_{\Sigma;H,y:[\sigma_2/\alpha]\sigma}) \\ &= \mathbf{curry}([K][\![N]\!]_{\Sigma,\alpha;H,y:\sigma}) \qquad \text{(hyp)} \\ &= [K](\mathbf{curry}([\![N]\!]_{\Sigma,\alpha;H,y:\sigma})) \qquad \text{(Lemma 5.12.1)} \\ &= [K]t. \end{aligned}$$

(3) $M \equiv \Lambda\beta.N$. Suppose that $\sigma_1 = \prod\beta.\sigma$ so that $H \vdash_{\Sigma,\alpha,\beta} N : \sigma$. Then

$$\begin{aligned} s &= [\![\Lambda\beta.[\sigma_2/\alpha]N]\!]_{\Sigma;H} \\ &= \mathbf{Curry}([\![[\sigma_2/\alpha]N]\!]_{\Sigma,\beta;H}) \\ &= \mathbf{Curry}([K \circ \mathbf{Fst}_{(\mathbf{D}^{\mathrm{EP}})^m,\mathbf{D}^{\mathrm{EP}}}][\![N]\!]_{\Sigma,\beta,\alpha;H}) \qquad \text{(hyp)} \\ &= [K](\mathbf{Curry}([\![N]\!]_{\Sigma,\beta;H})) \qquad \text{(Lemmas 5.12.2 and 5.14)} \\ &= [K]t. \end{aligned}$$

(4) $M \equiv N_1(N_2)$. Suppose that $H \vdash_{\Sigma,\alpha} N_1 : \sigma \to \sigma_1$ so that $H \vdash_{\Sigma,\alpha} N_2 : \sigma$. Then

$$\begin{aligned} s &= [\![([\sigma_2/\alpha]N_1)([\sigma_2/\alpha]N_2)]\!]_{\Sigma;H} \\ &= \mathbf{apply}([\![[\sigma_2/\alpha]N_1]\!]_{\Sigma;H}, [\![[\sigma_2/\alpha]N_2]\!]_{\Sigma;H}) \\ &= \mathbf{apply}([K][\![N_1]\!]_{\Sigma,\alpha;H}, [K][\![N_2]\!]_{\Sigma,\alpha;H}) \qquad \text{(hyp)} \\ &= [K](\mathbf{apply}([\![N_1]\!]_{\Sigma,\alpha;H}, [\![N_2]\!]_{\Sigma,\alpha;H})) \qquad \text{(Lemma 5.12.3)} \\ &= [K]t. \end{aligned}$$

(5) $M \equiv N\{\sigma\}$. Suppose that $\sigma_1 = [\sigma/\alpha_1]\tau$ so that $H \vdash_{\Sigma,\alpha} N : \prod\alpha_1.\tau$. Then

$$\begin{aligned}
s &= [\![([\sigma_2/\alpha]N)\{[\sigma_2/\alpha]\tau\}]\!]_{\Sigma;H} \\
&= \mathbf{Apply}([\![[[\sigma_2/\alpha]N]\!]_{\Sigma;H}, [\![[\sigma_2/\alpha]\sigma]\!]_{\Sigma}) \\
&= \mathbf{Apply}([K][\![N]\!]_{\Sigma,\alpha;H}, [\![[\sigma_2/\alpha]\sigma]\!]_{\Sigma}) \qquad \text{(hyp)} \\
&= \mathbf{Apply}([K][\![N]\!]_{\Sigma,\alpha;H}, [\![\sigma]\!]_{\Sigma,\alpha} \circ < \mathbf{Id}_{(\mathbf{D}^{\mathrm{EP}})^m}, K >)) \qquad \text{(Lemma 5.16)} \\
&= [K](\mathbf{Apply}([\![N]\!]_{\Sigma,\alpha;H}, [\![\sigma]\!]_{\Sigma,\alpha}) \qquad \text{(Lemma 5.12.4)} \\
&= [K]t. \quad \square
\end{aligned}$$

Lemma. (5.18) *Suppose $H \vdash_{\Sigma} M : \sigma_1 \to \sigma_2$. If x does not appear in H, then*

$$[\![M]\!]_{\Sigma;H,x:\ \sigma_1} = [\![M]\!]_{\Sigma;H} \circ \mathbf{fst}_{n,1}.$$

Proof. By structural induction on M. $\square$

Lemma. (5.19) *Suppose $H \vdash_{\Sigma} M : \sigma$. If α does not belong to Σ, then*

$$[\![M]\!]_{\Sigma,\alpha;H} = [\![M]\!]_{\Sigma;H} \circ \mathbf{Fst}_{(\mathbf{D}^{\mathrm{EP}})^m,\mathbf{D}^{\mathrm{EP}}}.$$

Proof. By structural induction on M. $\square$

Now we finally can prove the Soundness theorem:

Theorem. (5.20) (Soundness) *Equational rules for second order lambda calculus are satisfied under the given interpretation.*

Proof. Although there are eleven rules, the proofs of most of them are immediate. The only non-trivial proofs are those of the rules β, type β, η and type η.

(1) β rule:

$$\frac{H, x : \sigma_1 \vdash_{\Sigma} M : \sigma_2 \quad H \vdash_{\Sigma} N : \sigma_1}{H \vdash_{\Sigma} (\lambda x : \sigma_1.M)(N) = [N/x]M : \sigma_2}$$

This rule immediately follows from the lemma 5.15.

(2) type β rule:

$$\frac{H \vdash_{\Sigma,\alpha} M : \sigma_1}{H \vdash_{\Sigma} (\Lambda\alpha.M)\{\sigma_2\} = [\sigma_2/\alpha]M : [\sigma_2/\alpha]\sigma_1}$$

This rule immediately follows from the lemma 5.17.

(3) η rule:

$$\frac{H \vdash_{\Sigma} M : \sigma_1 \to \sigma_2}{H \vdash_{\Sigma} \lambda x : \sigma_1.M(x) = M : \sigma_1 \to \sigma_2}$$

where we have a restriction that x does not appear in H. We have

$$\begin{aligned}
[\![\lambda x:\sigma_1.M(x)]\!]_{\Sigma;H} &= \mathbf{curry}([\![M(x)]\!]_{\Sigma;H,x:\ \sigma_1}) \\
&= \mathbf{curry}(\mathbf{apply}([\![M]\!]_{\Sigma;H,x:\ \sigma_1}, \mathbf{snd}_{n,1})) \\
&= \mathbf{curry}(\mathbf{apply}([\![M]\!]_{\Sigma;H} \circ \mathbf{fst}_{n,1}, \mathbf{snd}_{n,1})) \qquad \text{(Lemma 5.18)} \\
&= [\![M]\!]_{\Sigma;H}.
\end{aligned}$$

(4) type η rule:

$$\frac{H \vdash_\Sigma M : \prod \alpha.\sigma}{H \vdash_\Sigma \Lambda\alpha.M\{\alpha\} = M : \prod \alpha.\sigma}$$

where α does not appear in Σ. Then

$$\begin{aligned}
&[\![\Lambda\alpha.M\{\alpha\}]\!]_{\Sigma;H} \\
&\quad = \mathbf{Curry}([\![M\{\alpha\}]\!]_{\Sigma,\alpha;H}) \\
&\quad = \mathbf{Curry}(\mathbf{Apply}([\![M]\!]_{\Sigma,\alpha;H}, [\![\alpha]\!]_{\Sigma,\alpha})) \\
&\quad = \mathbf{Curry}(\mathbf{Apply}([\![M]\!]_{\Sigma;H} \circ \mathbf{Fst}_{(\mathbf{D}^{\mathrm{EP}})^m,\mathbf{D}^{\mathrm{EP}}}, \mathbf{Snd}_{(\mathbf{D}^{\mathrm{EP}})^m,\mathbf{D}^{\mathrm{EP}}})) \quad \text{(Lemma 5.19)} \\
&\quad = [\![M]\!]_{\Sigma;H}. \quad \square
\end{aligned}$$

Remarks.

(1) We have presented the model in a way which slightly differs from the general definition of a model given in section 4.2. If we want to present the model in the Seely's formalization, we could proceed as follows. For the category **S** we take the category of locally finitely presentable categories (actually it would be enough to take the category based on the natural numbers which we used in section 4.3). This is cartesian closed. Moreover the functors from one locally finitely presentable category to another (defined up to isomorphism) form a set (and a cartesian closed category). The rest of the construction is the construction given in this chapter.

(2) For the construction was essential the fact that any Scott domain is a directed colimit of finite domains which **embed** into it - this follows from the fact that it is algebraic. We also, of course, need the domain to be a cpo. So the only feature of a Scott domain we didn't need is bounded completeness (and it is really possible to perform this construction in the category of algebraic cpos).

References

[**Asperti and Longo 91**] A. Asperti, G. Longo, *Categories, types, and structures: an introduction to category theory for the working computer scientist,* The MIT Press, Cambridge, Massachusetts, 1991.

[**Bainbridge et al. 90**] E. S. Bainbridge, P. J. Freyd, A. Scedrov, P. J. Scott, *Functorial polymorphism,* Theoret. Comput. Sci. **70** (1990), 35–64.

[**Bruce et al. 90**] K. B. Bruce, A. R. Meyer, J. C. Mitchell, *The semantics of second-order lambda calculus,* Inform. and Comput. **85** (1990), 76–134.

[**Coquand 89**] T. Coquand, *Categories of embeddings,* Theoret. Comput. Sci. **68** (1989), 221–237.

[**Coquand and Ehrhard 87**] T. Coquand, T. Ehrhard, *An equational presentation of higher-order logic,* Category Theory and Computer Science, LNCS 283, Springer-Verlag, Berlin, 1987, 40–56.

[**Coquand te al. 87**] T. Coquand, C. Gunter, G. Winskel, *dI-domains as a model of polymorphism,* Proceedings, Third Workshop on the Mathematical Foundations of Programming Language Semantics, LNCS 298, Springer-Verlag, New York, 1987.

[**Coquand te al. 89**] T. Coquand, C. Gunter, G. Winskel, *Domain theoretic models,* Inform. and Comput. **81** (1989), 123–167.

[**Fortune et al. 83**] S. Fortune, D. Leivant, M. O'Donnell, *The expressiveness of simple and secodn-order type structures,* Journal Assoc. Comp. Mach. **30** (1983), 151–185.

[**Girard 86**] J. Y. Girard, *The system F of variable types, fifteen years later,* Theoret. Comput. Sci. **45** (1986), 159–192.

[**Girard et al. 89**] J. Y. Girard, P. Taylor, Y. Lafont, *Proofs and Types,* Cambridge University Press, Cambridge, 1989.

[**Lambek and Scott 86**] J. Lambek, P. J. Scott, *Introduction to higher order categorical logic,* Cambridge University Press, Cambridge, 1986.

[**Lawvere 69**] F. W. Lawvere, *Adjointness in foundations,* Dialectica **23** (1969), 281–296.

[**McCracken 79**] N. McCracken, *An investigation of a programming language with a polymorphic type structure,* PhD thesis, Syracuse University, 1979.

[**Pitts 87**] A. M. Pitts, *Polymorphism is set theoretic, constructively,* Category Theory and Computer Science, LNCS 283, Springer-Verlag, Berlin, 1987, 12–39.

[**Reynolds 84**] J. C. Reynolds, *Polymorphism is not set-theoretic,* Semantics of Data Types, LNCS 173, Springer-Verlag, New York, 1984.

[**Robinson 91**] E. Robinson, *Notes on the second-order lambda calculus,* Notes for a logic for IT weekend course, Leeds, 1991.

[**Seely 87**] R. A. G. Seely, Categorical semantics for higher order polymorphic lambda calculus, Journal Symb. Logic **52** (1987), 969–989.