

**MODULAR CORRESPONDENCE BETWEEN
DEPENDENT TYPE THEORIES AND
CATEGORICAL UNIVERSES**

M. MAIETTI

REPORT No. 44, 2000/2001

ISSN 1103-467X

ISRN IML-R- -44-00/01- -SE



INSTITUT MITTAG-LEFFLER
THE ROYAL SWEDISH ACADEMY OF SCIENCES

Modular correspondence between dependent type theories and categorical universes.*

Maria Emilia Maietti

Dipartimento di Matematica Pura ed Applicata, Università di Padova
via G. Belzoni n.7, I-35131 Padova, Italy
e-mail: maietti@math.unipd.it

Abstract

Based on previous work, we give a modular correspondence between various categorical universes and their internal languages in terms of extensional dependent type theories à la Martin-Löf. Starting from lex categories, through regular ones we provide the internal languages of different kinds of pretopoi and topoi.

With respect to the internal languages already known for some of these categories like topoi, the novelty of these calculi is that formulas corresponding to subobjects can be regained as particular types equipped with proof-terms according to the isomorphism “propositions as mono types”, invisible in the previous languages.

MSC 2000: 03G30 03B15 18C50

Keywords: Categorical logic, topoi, Higher-order logic and type theory, Categorical semantics of formal languages.

Contents

1	Introduction	2
2	Categorical universes	5
3	Extensional dependent type theories	7
4	The modular correspondence categorical property/type constructor	23
5	The syntactic categories out of the type theories	24
5.1	Lex category	24
5.2	The disjoint coproduct	25
5.3	The natural numbers object	26
5.4	The list object	26
5.5	The image	27
5.6	The quotient of an equivalence relation	28
5.7	Right adjoints on subobjects	30
5.8	The right adjoint to the pullback functor	31
5.9	The subobject classifier	31
6	The categorical semantics	31

*This preprint contains part of the talk given at the Mittag-Leffler Institute, Stockholm, Sweden May 2001.

7	Dependent type theory as internal language.	34
8	Conclusions	36

1 Introduction

Set theory, category theory and type theory provide frameworks to develop mathematics and in particular constructive mathematics, in the sense that the underlying logic is intuitionistic. We are interested in relating the frameworks proposed in category theory with those in type theory.

To formalize intuitionistic mathematics in type theory we have Martin-Löf’s Intuitionistic Type Theory [Mar75, Mar95, NPS90], also called Constructive Set Theory or Constructive Type Theory, which is predicative and Coquand’s Calculus of Constructions [CG88], which is impredicative.

In category theory topoi or pretopoi can be thought as universes where to develop mathematics because they provide models for certain kinds of set theory. The main example is the elementary topos, introduced by Lawvere and Tierney starting from an axiomatization of a Grothendieck topos free of set-theoretic assumptions. In suitable topoi it is also possible to model fragments of classical set theory, like bounded Zermelo set theory [Col73, Mit72, MM92].

More recently, Joyal and Moerdijk explored how to provide models for the full Zermelo-Fraenkel set theory in a categorical setting [JM95] taking an Heyting pretopos with a natural numbers object as universe.

Particular pretopoi, called arithmetic universes, were introduced by André Joyal to give a categorical proof of Gödel incompleteness theorems, which he performed in the initial one [Wra85]. They are built through three stages, from a cartesian category with a parameterized natural numbers object, through a regular category, to a pretopos with parameterized list objects.

To relate the considered frameworks in category theory and type theory we can see at least two ways: we look either for categorical models of the considered type theories or for the internal type theory of the categorical universes.

Here, we pursue the second way, namely to look for the internal language of topoi, pretopoi and related universes in terms of a dependent type theory à la Martin-Löf. To find them we are facilitated from the fact that the correspondence between dependent type theories and the considered categories is modular in the sense that we can single out a precise correspondence between universal categorical properties and type-theoretic constructors.

Internal languages à la Martin-Löf prepare the grounds for a purely type-theoretical comparison between the considered categories and Martin-Löf’s type theory. The ultimate goal would be to put all these dependent type theories in a hierarchy and explore translations among them. We also think that the internal language could be a tool to transfer techniques from type theory to category theory and the way around.

In the literature connections between categorical universes and logic have been explored especially in terms of many-sorted logics, where the sorts are simple types.

Makkay and Reyes found that pretopoi can be characterized with respect to logical categories, which are the necessary structures to interpret many-sorted coherent logic [MR77], but, as far as we know, no explicit internal calculus for pretopoi has been proposed before ours.

About topoi, it is well known how to associate to any topos its Mitchell-Benabou internal language in a way as categorical objects correspond to types, morphisms to typed terms and subobjects to many-sorted formulas, which are terms of the subobject classifier [MM92, LS86]. One of the main characteristics of this language is the presence of two kinds of syntactic entities, types and formulas, both equipped with suitable judgements: about types the judgements for formation of types with corresponding terms and equalities and about formulas judgements in the form of usual sequents. There is no constructor turning formulas into types, namely no isomorphism of the sort “proposition as types” is visible: propositions are not equipped with proof-terms.

On the contrary, developing internal languages à la Martin-Löf we have only one syntactic entity: types with corresponding proof-terms and equalities. Nevertheless, to capture all the constructions of

topoi or pretopoi or even lex categories simple types, like those used in Mitchell-Benabou language, do not suffice and dependent types are crucial for that.

Using dependent types and thanks to the modularity of the correspondence universal categorical property/type constructor it is possible to get the internal language of various categories, from lex to regular categories, pretopoi, arithmetic universes, topoi and variations of them.

The link between dependent type theories and categories can be established once we know that the considered categories provide a complete categorical semantics for their corresponding dependent type theories. However, this check is only necessary but does not suffice to say that a dependent type theory provides also their internal language, since this requires to prove a sort of equivalence between the category of theories and the category of categories under consideration.

The categorical semantics of a dependent typed calculus is more complex than that of a simple typed calculus where types are interpreted as objects and terms as morphisms. Fixed the category \mathcal{C} where to interpret the calculus, the idea is to interpret a dependent type under a context as a sequence of morphisms of \mathcal{C} and a dependent term as a morphism of a suitable comma category of \mathcal{C} . Since we want to interpret substitution via pullback, then to overcome the well known coherence problems we use the split fibration associated to the codomain fibration of the category, putting together techniques developed in [Ben85, Hof94, Car86].

To interpret dependent types according to the above categorical semantics a category \mathcal{C} has to satisfy general conditions:

1. \mathcal{C} has to have finite limits
2. The structure of \mathcal{C} necessary to interpret the type constructors on closed types has to be local, i.e. for every object $A \in Ob\mathcal{C}$ the comma category \mathcal{C}/A enjoys the same structure of \mathcal{C} (for example, if \mathcal{C} is a topos then \mathcal{C}/A should be a topos for every $A \in Ob\mathcal{C}$). This is because a dependent type is interpreted in a suitable comma category which has to interpret all the type constructors under the same dependency.
3. The structure of \mathcal{C} has to be preserved by the pullback functor $f^* : \mathcal{C}/A \rightarrow \mathcal{C}/B$ for every morphism $f : B \rightarrow A$ of \mathcal{C} . This is because, if we interpret substitution via pullback, then we need the preservation under pullback to make the interpretation of type constructors closed under substitution.

Therefore, these become necessary conditions for a category to enjoy a dependent typed internal language.

Having in mind the above categorical semantics we can recognize how a dependent type theory à la Martin-Löf can describe the structure of subobjects avoiding the use of formulas and sequents. Indeed, it turns out that a mono type, which is defined as a type with at most one proof, i.e. formally a type $B(x) [\Gamma]$ for which we can derive

$$y = z \in B(x) [\Gamma, y \in B(x), z \in B(x)]$$

gets interpreted in a sequence of morphisms whose last one (which interprets the type itself, whilst the rest interprets the context) is mono. Hence, mono types correspond to monomorphisms. This is crucial to capture the subobject classifier of a topos or the quotients restricted to monic equivalence relations of a pretopos, only speaking about types.

More in details, universal categorical properties correspond to already known type constructors of Martin-Löf's type theory or new ones giving a modular correspondence between categories and dependent type theories. Finite limits correspond to the terminal type, the indexed sum types, the extensional equality types in [Mar84]. These are the basic module since pullbacks are necessary to interpret substitution and the equality types are the basic dependent types. Then, the right adjoint to the pullback functor between comma categories corresponds to the dependent product type in [Mar84, NPS90], the right adjoint to the pullback functor between subobjects to the dependent product type restricted to mono types and the left adjoint to the pullback functor between subobjects (or stable images) to the indexed sum type made mono and restricted to mono types. Stable quotients of mono equivalence relations correspond to extensional quotient types based only on mono type equivalence relations and effectiveness is expressed as an axiom. The stable initial object corresponds to the falsum type in

[Mar84, NPS90] and stable binary disjoint coproducts to the disjoint sum types in [Mar84, NPS90] with the addition of an axiom for disjointness. The parameterized natural numbers object corresponds to the natural numbers type in [Mar84, NPS90] and parameterized list objects to list types in [Mar84, NPS90]. Finally the subobject classifier corresponds to an extensional universe type encoding mono types up to equiprovability.

Another difference between internal languages à la Martin-Löf and those in the form of a many-sorted logic is that with the former we can build a syntactic category taking closed types as objects and terms as morphisms. In contrast, with the latter, for example in the case of building out the syntactic topos from Mitchell-Benabou language as in [LS86],[Bel88], closed terms of powersets, i.e. formulas, are taken as objects, and functional relations are taken as morphisms. But, only looking at the syntactic category built out from a dependent type theory, we can realize once more how monomorphisms corresponds to mono types. Indeed, the main advantage of using internal languages à la Martin-Löf is the possibility of making visible that considering formulas as subobjects yields to follow the isomorphism *propositions as mono types*, i.e. *types with at most one proof* or more generally *formulas as mono dependent types*, i.e. *dependent types with at most one proof*.

This isomorphism provides a key to compare Martin-Löf's Constructive Type Theory with categorical universes from a type-theoretical point of view.

If we look at these two frameworks only from the type perspective, one of the main difference is that all the internal dependent type theories of the so far considered categories are extensional according to the extensional version of Martin-Löf's type theory in [Mar84]. Indeed, for example, the internal dependent type theory of a topos includes the fragment without universes of Martin-Löf's type theory in [Mar84] as a subsystem. Though, what is nowadays considered the correct version of Martin-Löf's Constructive Type Theory is the intensional one in [NPS90, Mar75]. This means that, whilst in intensional Martin-Löf's Constructive Type Theory the judgemental equality between terms, corresponding to the rewrite equality, written $t = s : A [\Gamma]$, does not coincide with the propositional equality type, written $\text{ld}(A, t, s)$, with the consequence of remaining decidable, in the extensional dependent type theories the judgemental equality does coincide with the propositional equality type with the consequence of loosing in general its decidability. For sake of clearness, we must say that the internal language of the considered categories is necessarily extensional because the equality between morphisms in the syntactic category is defined as the judgemental equality and not as the propositional equality. However, the latter is not that suitable to establish a modular correspondence between universal categorical properties and known type-theoretic constructors, even for lex categories.

Looking at the two frameworks, Martin-Löf's Constructive Type Theory and categorical universes through their internal dependent typed languages, from the proposition perspective the things go very differently: in Martin-Löf's Constructive Type Theory the isomorphism *proposition as types* holds, while in the considered categorical universes, like topoi, *propositions as mono types* holds, as already said. One big consequence is soon visible: while Martin-Löf thinking of propositions as types arrives to conceive a strong existential quantifiers that yields to the propositional axiom of choice, topoi, being governed by propositions as mono types, admit only the usual existential quantifier with no internal existence property or witness of which it predicates and hence no propositional axiom of choice. However, in topoi we can derive the axiom of unique choice and by our internal dependent language we can see why: in this case the existential quantifier becomes equivalent to the strong one.

There are also constructions that in the presence of *proposition as mono types* do preserve constructivity, which do not in general with *propositions as types*:

1. in topoi we can have extensional powersets which does not destroy constructivity, whilst, considering subsets as general propositional functions we have that extensional powersets, even if added to an intensional framework, yield classical logic; this is proved mimicking Diaconescu's argument that in a topos the propositional axiom of choice implies the principle of excluded middle [MV99];
2. in pretopoi (and also topoi) we can have effective quotients; but effectiveness does not keep constructivity [Mai99a] if applied to generic relations instead of only to mono type relations as in the internal type theory of pretopoi and topoi.

A careful analysis of pros and cons of the two frameworks could also reveal some compromises

to weaken a topos to be constructively coherent with the axiom of choice or to extend Martin-Löf's Constructive Type Theory with stronger constructors closer to those of a topos as much as possible.

So far we have discussed how the internal languages à la Martin-Löf prepare the grounds for a type-theoretic comparison between categories and type theories. We also recall that an obvious use of the internal language of a category is to perform categorical proofs in a logical way. For example, we intend to use the internal language of an arithmetic universe to give a type-theoretic version of the proof of Gödel's incompleteness, done categorically by André Joyal within the initial arithmetic universe.

The work reported here is based on the PhD-thesis [Mai98b]. The internal language of Heyting pretopoi also appeared in [Mai98a].

2 Categorical universes

We proceed by recalling the definitions of the categorical universes we consider, starting with lex categories to end with topoi.

Def. 2.1 A **lex category** is a category with finite limits (or finitely complete category), i.e. with a terminal object, binary products and equalizers [Mac71].

We recall that for a category having a terminal object and pullbacks is equivalent to being finitely complete, and that having finite products is equivalent to having a terminal object and binary products.

Def. 2.2 A lex category is said **arithmetic** if it has a parameterized natural numbers object.

where

Def. 2.3 A **parameterized natural numbers object** in a category with finite products is an object N together with maps $0 : 1 \rightarrow N$, $s : N \rightarrow N$ such that for every $b : B \rightarrow Y$ and $g : Y \rightarrow Y$ there is a unique $rec(b, g)$ making the following diagrams commute

$$\begin{array}{ccccc}
 B & \xrightarrow{\langle id, 0 \cdot !_B \rangle} & B \times N & \xleftarrow{id \times s} & B \times N \\
 & \searrow b & \downarrow rec(b, g) & & \downarrow rec(b, g) \\
 & & Y & \xleftarrow{g} & Y
 \end{array}$$

with $!_B : B \rightarrow 1$ the unique map towards the terminal object.

It is worth to recall here that in presence of function spaces (or exponentials), like in a cartesian closed category (see [MM92, LS86] for a definition), this parameterized version of natural numbers object is equivalent to the usual natural numbers object.

Def. 2.4 A **regular category** is a finitely complete category with stable images [Jac99, Tay97].

Def. 2.5 A **distributive category** is a finitely complete category with stable finite disjoint coproducts [Coc90].

Def. 2.6 A **locos** is a distributive category with parameterized list objects [Coc90].

where parameterized list objects are defined as follows:

Def. 2.7 A finitely complete category \mathcal{C} has **parameterized list objects** if for all objects $A \in Ob\mathcal{C}$, there is an object $List(A)$ with maps $r_o^A : 1 \rightarrow List(A)$, $r_1^A : List(A) \times A \rightarrow List(A)$ such that for every $b : B \rightarrow Y$ and $g : Y \times A \rightarrow Y$ there is a unique $rec_l(b, g)$ making the following diagrams commute

$$\begin{array}{ccccc}
 B & \xrightarrow{r_o^A} & B \times List(A) & \xleftarrow{id \times r_1^A} & B \times (List(A) \times A) \\
 & \searrow b & \downarrow rec_l(b, g) & & \downarrow (rec_l(b, g) \times id_A) \cdot \sigma \\
 & & Y & \xleftarrow{g} & Y \times A
 \end{array}$$

where $\sigma : B \times (List(A) \times A) \rightarrow (B \times List(A)) \times A$ is the associative isomorphism $\langle \langle \pi_1, \pi_1 \cdot \pi_2 \rangle, \pi_2 \cdot \pi_2 \rangle$.

In [Coc90] there is an equivalent definition of finitely complete categories with list objects (also called list-arithmetic lex categories) in terms of recursive objects and preservation of recursive objects by the pullback functor $!_D^* : \mathcal{C} \rightarrow \mathcal{C}/D$ sending an object B to $\pi_1 : D \times B \rightarrow D$. Finally, we recall the categorical definition of pretopos, locally cartesian closed category, topos and variations of them:

Def. 2.8 A **pretopos** is a category equipped with finite limits, stable finite disjoint sums and stable effective quotients of equivalence relations [MR77, JM95].

Def. 2.9 A **Heyting pretopos** is a pretopos where the pullback functor on subobjects has a right adjoint [JM95].

Def. 2.10 An **arithmetic universe** is a pretopos with parameterized list objects [Mai99b].

Def. 2.11 A **locally cartesian closed category** is a category equipped with finite limits and right adjoints to pullback functors [Jac99, Tay97].

Def. 2.12 A **topos** is a category equipped with finite limits, exponentials and a subobject classifier [MR77, MM92].

Since the aim of this note is to describe the internal dependent type theory of the categories mentioned so far, we list necessary conditions that a category \mathcal{C} has to satisfy to enjoy a dependent typed internal language:

1. \mathcal{C} has to be finitely complete.
2. The structure of \mathcal{C} necessary to interpret the type constructors on closed types has to be *local*, i.e. for every object $A \in \text{Ob}\mathcal{C}$ the comma category \mathcal{C}/A enjoys the same structure of \mathcal{C} (for example, if \mathcal{C} is a topos then \mathcal{C}/A should be a topos for every $A \in \text{Ob}\mathcal{C}$). This is because a dependent type is interpreted in a suitable comma category which has to interpret all the type constructors under the same dependency.
3. The structure of \mathcal{C} has to be *preserved by the pullback functor* $f^* : \mathcal{C}/A \rightarrow \mathcal{C}/B$ for every morphism $f : B \rightarrow A$ of \mathcal{C} . This is because, if we interpret substitution via pullback, then we need the preservation under pullback to make the interpretation of type constructors closed under substitution.

The categories mentioned so far satisfy the necessary conditions to enjoy an internal dependent type theory:

Proposition 2.13 *Lex, lex arithmetic, regular, distributive categories, locoi, pretopoi, arithmetic universes, Heyting pretopoi, locally cartesian closed categories and topoi are local and their structures are preserved by pullbacks in the above sense.*

Proof. The proof of this statement is modular on the universal categorical properties such categories enjoy. The local property of a finitely complete category follows from the fact that in \mathcal{C}/A the identity on A is the terminal object and that the forgetful functor $U : \mathcal{C}/A \rightarrow \mathcal{C}$ creates pullbacks. Moreover, the above forgetful functor creates colimits and essentially by definition we have that for every $f : A \rightarrow B$ the pullback functor $f^* : \mathcal{C}/B \rightarrow \mathcal{C}/A$ preserves finite coproducts and quotients considering that, given an equivalence relation in \mathcal{C}/D

$$\begin{array}{ccc} R & \xrightarrow{\rho} & A \times_D A \\ & \searrow r & \swarrow a \cdot \pi_1 \\ & & D \end{array}$$

$\langle \pi_1 \cdot \rho, \pi_2 \cdot \rho \rangle : R \rightarrow A \times A$ is also an equivalence relation in \mathcal{C} , where $\pi_i : A \times_D A \rightarrow A$ for $i : 1, 2$ are the projections of the pullback of $a : A \rightarrow D$ along itself in \mathcal{C} . These considerations suffice to see that a pretopos is local. Then, it is easy to check that for every object A of a category \mathcal{C} with products, a parameterized natural numbers object in \mathcal{C}/A is $\pi_1 : A \times \mathcal{N} \rightarrow A$, where \mathcal{N} is a natural object of \mathcal{C} .

An Heyting pretopos \mathcal{P} is local because the subobjects in the comma category correspond to subobjects in \mathcal{P} . Then, to check that right adjoints are stable under pullbacks means that Beck-Chevalley conditions are satisfied, which also follows.

For list objects the local property is more delicate and we know that lists are local only starting from *locoi* (locality of *locoi* is also stated in [Coc90]). In our proof we assume to know the internal dependent type theory of a finitely complete distributive category. We get then the internal type theory of a *locos* simply adding list types restricted to closed types, which are interpreted as parameterized list objects. By means of this internal language we show how to build list types on arbitrary dependent types, corresponding to list-objects in a slice category, and that in the suitable slice syntactic category they are stable under pullbacks.

Finally, the proof that a topos is local can be found in [MM92] or [Joh77].

■

We stress that in order to be able to interpret a typed calculus a given choice of the considered categorical constructors needs to be made since they are usually defined up to isomorphism. However, for dependent typed calculi this is not enough to give a direct interpretation in the category, since we need further properties that the choice made has to satisfy and canonical choices satisfying such properties are in general not known. For example, interpreting substitution via pullback we need some functoriality of pullbacks and even for the category of sets and functions *Set* we do not know any canonical functorial choice of pullbacks. To overcome this difficulty we will make use of the machinery of fibred functors [Jac99].

3 Extensional dependent type theories

We start with the description of extensional dependent typed calculi following Martin-Löf's extensional type theory [Mar84]

Any typed system is equipped with types, which should be thought of as sets or data types, and with typed terms which represent proofs of the types to which they belong. To speak about them in the style of Martin-Löf's type theory, we have four kinds of judgements [NPS90]:

$$A \text{ type} \quad A = B \quad a \in A \quad a = b \in A$$

that is the judgements about type formation and their terms, the equality between types and the equality between terms of the same type.

The contexts of these judgements are telescopic [dB91], since types are allowed to depend on variables of other types. The contexts are generated by the following rules

$$1C) \quad \emptyset \text{ cont} \quad 2C) \quad \frac{\Gamma \text{ cont} \quad A \text{ type } [\Gamma]}{\Gamma, x \in A \text{ cont}} \quad (x \in A \notin \Gamma)$$

plus the rules of equality between contexts [Str91], [Pit95]. In the following, we present the inference rules to construct type judgements and term judgements with their equality judgements by recursion. One should also add all the inference rules that express reflexivity, symmetry and transitivity of the equality between types and terms and the set equality rule

$$\text{conv}) \quad \frac{a \in A [\Gamma] \quad A = B [\Gamma]}{a \in B [\Gamma]}$$

Moreover, by the following rule we assume typed variables

$$\text{var}) \quad \frac{\Gamma, x \in A, \Delta \text{ cont}}{x \in A [\Gamma, x \in A, \Delta]}$$

We can derive the structural rules of weakening, substitution and of a suitable exchange.

Now, we give the formation rule for types specific to the various calculi and then the introduction, elimination and conversion rules of their terms. Beside them we should add the corresponding formation equality rules for types and introduction and elimination equality rules for terms as in [Mar84], but to be short we omit them.

We adopt the usual definitions of bound and free occurrences of variables and we identify two terms under α -conversion.

Remark 3.1 In the following, the piece of context common to all judgements involved in a rule will be omitted. The typed variables appearing in a context are meant to be added to the implicit context as the last one.

The calculus for lex categories $\overline{\mathcal{T}}_{lex}$	
Terminal type	
$\frac{}{\top \text{ type}} \text{Tr}$	$\frac{}{\star \in \top} \text{I-Tr}$
$\frac{t \in \top}{t = \star \in \top} \text{C-Tr}$	
Indexed Sum type	
$\frac{C(x) \text{ type } [x \in B]}{\Sigma_{x \in B} C(x) \text{ type}} \Sigma$	$\frac{b \in B \quad c \in C(b)}{\langle b, c \rangle \in \Sigma_{x \in B} C(x)} \text{I-}\Sigma$
$\frac{M(z) \text{ type } [z \in \Sigma_{x \in B} C(x)] \quad d \in \Sigma_{x \in B} C(x) \quad m(x, y) \in M(\langle x, y \rangle) [x \in B, y \in C(x)]}{\text{split}(d, m) \in M(d)} \text{E-}\Sigma$	
$\frac{M(z) \text{ type } [z \in \Sigma_{x \in B} C(x)] \quad b \in B \quad c \in C(b) \quad m(x, y) \in M(\langle x, y \rangle) [x \in B, y \in C(x)]}{\text{split}(\langle b, c \rangle, m) = m(b, c) \in M(\langle b, c \rangle)} \text{C-}\Sigma$	
Extensional Equality type	
$\frac{C \text{ type } \quad c \in C \quad d \in C}{\text{Eq}(C, c, d) \text{ type}} \text{Eq}$	$\frac{c \in C}{\text{eq}_C(c) \in \text{Eq}(C, c, c)} \text{I-Eq}$
$\frac{p \in \text{Eq}(C, c, d)}{c = d \in C} \text{E-Eq}$	$\frac{p \in \text{Eq}(C, c, d)}{p = \text{eq}_C(c) \in \text{Eq}(C, c, d)} \text{C-Eq}$

The calculus of arithmetic lex categories \mathcal{T}_{alex}
 =
 Lex calculus

+

Natural Numbers type

$\overline{N \text{ type}} \text{ nat}$ $\overline{0 \in N} \text{ I}_1\text{-nat}$ $\frac{n \in N}{s(n) \in N} \text{ I}_2\text{-nat}$

$\frac{\begin{array}{l} L(z) \text{ type } [z \in N] \\ n \in N \quad a \in L(0) \quad l(x, y) \in L(s(x)) \quad [x \in N, y \in L(x)] \end{array}}{\text{Rec}(a, l, n) \in L(n)} \text{ E-nat}$

$\frac{\begin{array}{l} L(z) \text{ type } [z \in N] \\ a \in L(0) \quad l(x, y) \in L(s(x)) \quad [x \in N, y \in L(x)] \end{array}}{\text{Rec}(a, l, 0) = a \in L(0)} \text{ C}_1\text{-nat}$

$\frac{\begin{array}{l} L(z) \text{ type } [z \in N] \\ n \in N \quad a \in L(0) \quad l(x, y) \in L(s(x)) \quad [x \in N, y \in L(x)] \end{array}}{\text{Rec}(a, l, s(n)) = l(n, \text{Rec}(a, l, n)) \in L(s(n))} \text{ C}_2\text{-nat}$

The calculus of regular categories \mathcal{T}_{reg}

=

Lex calculus

+

Mono Existential type

$$\frac{C(x) \text{ type } [x \in B] \quad y = z \in C(x) [x \in B, y \in C(x), z \in C(x)]}{\exists_{x \in B} C(x) \text{ type}} \exists \quad \frac{\exists_{x \in B} C(x) \text{ type} \quad b \in B \quad c \in C(b)}{(b, c) \in \exists_{x \in B} C(x)} \text{I-}\exists$$

$$\frac{\exists_{x \in B} C(x) \text{ type} \quad b \in B \quad c \in C(b) \quad d \in B \quad t \in C(d)}{(b, c) = (d, t) \in \exists_{x \in B} C(x)} \text{eq-}\exists$$

$$\frac{M(z) \text{ type } [z \in \exists_{x \in B} C(x)] \quad y = z \in M(w) [w \in \exists_{x \in B} C(x), y \in M(w), z \in M(w)] \quad d \in \exists_{x \in B} C(x) \quad m(x, y) \in M((x, y)) [x \in B, y \in C(x)]}{\text{Ex}(d, m) \in M(d)} \text{E-}\exists$$

or

+

Quotient types on terminal type

$$\frac{A \text{ type}}{A/\top \text{ type}} \text{Qtr}$$

$$\frac{a \in A}{[a] \in A/\top} \text{I-Qtr} \quad \frac{a \in A \quad b \in A}{[a] = [b] \in A/\top} \text{eq-Qtr}$$

$$\frac{L(z) \text{ type } [z \in A/\top] \quad p \in A/\top \quad l(x) \in L([x]) [x \in A] \quad l(x) = l(y) \in L([x]) [x \in A, y \in A]}{\text{Q}(l, p) \in L(p)} \text{E-Qtr}$$

$$\frac{L(z) \text{ type } [z \in A/\top] \quad a \in A \quad l(x) \in L([x]) [x \in A] \quad l(x) = l(y) \in L([x]) [x \in A, y \in A]}{\text{Q}(l, [a]) = l(a) \in L([a])} \text{C-Qtr}$$

The calculus of distributive categories \mathcal{T}_{dis}

=

Lex calculus

+

Disjoint Sum type

$$\frac{C \text{ type} \quad D \text{ type}}{C + D \text{ type}} + \frac{c \in C}{\text{inl}(c) \in C + D} I_{1-+} \quad \frac{d \in D}{\text{inr}(c) \in C + D} I_{2-+}$$

$$\frac{A(z) \text{ type } [z \in C + D] \quad w \in C + D \quad a_C(x) \in A(\text{inl}(x)) [x \in C] \quad a_D(y) \in A(\text{inr}(y)) [y \in D]}{\mathcal{D}(w, a_C, a_D) \in A(w)} E_{-+}$$

$$\frac{A(z) \text{ type } [z \in C + D] \quad c \in C \quad a_C(x) \in A(\text{inl}(x)) [x \in C] \quad a_D(y) \in A(\text{inr}(y)) [y \in D]}{\mathcal{D}(\text{inl}(c), a_C, a_D) = a_C(c) \in A(\text{inl}(c))} C_{1-+}$$

$$\frac{A(z) \text{ type } [z \in C + D] \quad d \in D \quad a_C(x) \in A(\text{inl}(x)) [x \in C] \quad a_D(y) \in A(\text{inr}(y)) [y \in D]}{\mathcal{D}(\text{inr}(d), a_C, a_D) = a_D(d) \in A(\text{inr}(d))} C_{2-+}$$

False type

$$\frac{}{\perp \text{ type}} F_s \quad \frac{a \in \perp \quad A \text{ type}}{r_o(a) \in A} E\text{-}F_s$$

Disjointness

$$\frac{c \in C \quad d \in D \quad \text{inl}(c) = \text{inr}(d) \in C + D}{m(c, d) \in \perp}$$

The calculus of locoi \mathcal{T}_{loc}

=

Distributive calculus

+

List type

$$\frac{C \text{ type}}{List(C) \text{ type}} \text{ list} \quad \frac{}{\epsilon \in List(C)} \text{ I}_1\text{-list} \quad \frac{s \in List(C) \quad c \in C}{\text{cons}(s, c) \in List(C)} \text{ I}_2\text{-list}$$

$$\frac{\begin{array}{c} L(z) \text{ type } [z \in List(C)] \\ s \in List(C) \quad a \in L(\epsilon) \quad l(x, y, z) \in L(\text{cons}(x, y)) \quad [x \in List(C), y \in C, z \in L(x)] \end{array}}{\text{Rec}_l(a, l, s) \in L(s)} \text{ E-list}$$

$$\frac{\begin{array}{c} L(z) \text{ type } [z \in List(C)] \\ s \in List(C) \quad a \in L(\epsilon) \quad l(x, y, z) \in L(\text{cons}(x, y)) \quad [x \in List(C), y \in C, z \in L(x)] \end{array}}{\text{Rec}_l(a, l, \epsilon) = a \in L(\epsilon)} \text{ C}_1\text{-list}$$

$$\frac{\begin{array}{c} L(z) \text{ type } [z \in List(C)] \\ s \in List(C) \quad c \in C \quad a \in L(\epsilon) \quad l(x, y, z) \in L(\text{cons}(x, y)) \quad [x \in List(C), y \in C, z \in L(x)] \end{array}}{\text{Rec}_l(a, l, \text{cons}(s, c)) = l(s, c, \text{Rec}_l(a, l, s)) \in L(\text{cons}(s, c))} \text{ C}_2\text{-list}$$

The calculus of pretopoi \mathcal{T}_{ptop}
=
Distributive calculus

+

Quotient type

$R(x, y)$ type $[x \in A, y \in A]$, $z = w \in R(x, y)[x \in A, y \in A, z \in R(x, y), w \in R(x, y)]$
 $c_1 \in R(x, x)[x \in A]$, $c_2 \in R(y, x)[x \in A, y \in A, z \in R(x, y)]$
 $c_3 \in R(x, z)[x \in A, y \in A, z \in A, w \in R(x, y), w' \in R(y, z)]$

A/R type

Q

$$\frac{a \in A \ A/R \ type}{[a] \in A/R} \text{ I-Q} \quad \frac{a \in A \ b \in A \ d \in R(a, b)}{[a] = [b] \in A/R \ A/R \ type} \text{ eq-Q}$$

$$\frac{L(z) \ type \ [z \in A/R] \quad p \in A/R \quad l(x) \in L([x]) \ [x \in A] \quad l(x) = l(y) \in L([x]) \ [x \in A, y \in A, d \in R(x, y)]}{Q(l, p) \in L(p)} \text{ E-Q}$$

$$\frac{L(z) \ type \ [z \in A/R] \quad a \in A \quad l(x) \in L([x]) \ [x \in A] \quad l(x) = l(y) \in L([x]) \ [x \in A, y \in A, d \in R(x, y)]}{Q(l, [a]) = l(a) \in L([a])} \text{ C-Q}$$

Effectiveness

$$\frac{a \in A \ b \in A \quad [a] = [b] \in A/R}{f(a, b) \in R(a, b)}$$

$$\begin{array}{c}
\text{The calculus of Heyting pretopoi } \mathcal{T}_{hptop} \\
= \\
\text{Pretopos calculus} \\
+ \\
\text{Forall type} \\
\frac{C(x) \text{ type}[x \in B] \quad y = z \in C(x) \quad [x \in B, y \in C(x), z \in C(x)]}{\forall_{x \in B} C(x) \text{ type}} \forall \\
\frac{c \in C(x)[x \in B] \quad y = z \in C(x) \quad [x \in B, y \in C(x), z \in C(x)]}{\lambda x^B. c \in \forall_{x \in B} C(x)} \text{I-}\forall \\
\frac{b \in B \quad f \in \forall_{x \in B} C(x)}{\text{Ap}(f, b) \in C(b)} \text{E-}\forall \\
\frac{f \in \forall_{x \in B} C(x)}{\lambda x^B. \text{Ap}(f, x) = f \in \forall_{x \in B} C(x)} \eta\text{C-}\forall
\end{array}$$

In the η -equality for the forall type the variable x^B does not appear free in f . Note also that its β -equality rule becomes derivable for the forall type, since the argument type is inhabited with at most one proof.

$$\begin{array}{c}
\text{The calculus of arithmetic universes } \mathcal{T}_{au} \\
\text{(or pretopoi with parameterized list objects)} \\
= \\
\text{Pretopos calculus} \\
+ \\
\text{List type} \\
\frac{C \text{ type}}{\text{List}(C) \text{ type}} \text{list} \quad \frac{}{\epsilon \in \text{List}(C)} \text{I}_1\text{-list} \quad \frac{s \in \text{List}(C) \quad c \in C}{\text{cons}(s, c) \in \text{List}(C)} \text{I}_2\text{-list} \\
\frac{s \in \text{List}(C) \quad L(z) \text{ type } [z \in \text{List}(C)] \quad a \in L(\epsilon) \quad l(x, y, z) \in L(\text{cons}(x, y)) \quad [x \in \text{List}(C), y \in C, z \in L(x)]}{\text{Rec}_1(a, l, s) \in L(s)} \text{E-list} \\
\frac{s \in \text{List}(C) \quad L(z) \text{ type } [z \in \text{List}(C)] \quad a \in L(\epsilon) \quad l(x, y, z) \in L(\text{cons}(x, y)) \quad [x \in \text{List}(C), y \in C, z \in L(x)]}{\text{Rec}_1(a, l, \epsilon) = a \in L(\epsilon)} \text{C}_1\text{-list} \\
\frac{s \in \text{List}(C) \quad L(z) \text{ type } [z \in \text{List}(C)] \quad c \in C \quad a \in L(\epsilon) \quad l(x, y, z) \in L(\text{cons}(x, y)) \quad [x \in \text{List}(C), y \in C, z \in L(x)]}{\text{Rec}_1(a, l, \text{cons}(s, c)) = l(s, c, \text{Rec}_1(a, l, s)) \in L(\text{cons}(s, c))} \text{C}_2\text{-list}
\end{array}$$

$$\begin{array}{c}
\text{The calculus of locally cartesian closed categories } \mathcal{T}_{lcc} \\
= \\
\text{The first order fragment of Martin-Löf's extensional type theory} \\
= \\
\text{Lex calculus} \\
+ \\
\text{Product type} \\
\frac{C(x) \text{ type } [x \in B]}{\prod_{x \in B} C(x) \text{ type}} \mathbf{F}\text{-}\Pi \\
\frac{c \in C(x)[x \in B]}{\lambda x^B . c \in \prod_{x \in B} C(x)} \mathbf{I}\text{-}\Pi \quad \frac{b \in B \quad f \in \prod_{x \in B} C(x)}{\text{Ap}(f, b) \in C(b)} \mathbf{E}\text{-}\Pi \\
\frac{b \in B \quad c \in C(x)[x \in B]}{\text{Ap}(\lambda x^B . c, b) = c(b) \in C(b)} \beta\mathbf{C}\text{-}\Pi \quad \frac{f \in \prod_{x \in B} C(x)}{\lambda x^B . \text{Ap}(f, x) = f \in \prod_{x \in B} C(x)} \eta\mathbf{C}\text{-}\Pi
\end{array}$$

$$\begin{array}{c}
\text{The calculus of topoi } \mathcal{T}_{top} \\
= \\
\text{Locally cartesian closed calculus} \\
+ \\
\text{Omega type} \\
\frac{}{\Omega \text{ type}} \Omega \quad \frac{B \text{ type} \quad y = z \in B [y \in B, z \in B]}{\{B\} \in \Omega} \mathbf{I}\text{-}\Omega \\
\frac{B \text{ type} \quad y = z \in B [y \in B, z \in B]}{C \text{ type} \quad y = z \in C [y \in C, z \in C]} \\
\frac{f \in B \leftrightarrow C}{\{B\} = \{C\} \in \Omega} \mathbf{eq}\text{-}\Omega \\
\frac{B \text{ type} \quad y = z \in B [y \in B, z \in B]}{\langle r_B, r_B^{-1} \rangle \in \text{Eq}(\Omega, \{B\}, \{\top\}) \leftrightarrow B} \beta\mathbf{C}\text{-}\Omega \quad \frac{q \in \Omega}{\{\text{Eq}(\Omega, q, \{\top\})\} = q \in \Omega} \eta\mathbf{C}\text{-}\Omega
\end{array}$$

From now on we shall often omit the word *type* in the type judgements. In the following, given a judgement $b(x) \in B(x)[x \in A]$ by the expression

$$(x)b(x)$$

we mean the equivalence class of $b(x) \in B(x)[x \in A]$ under the following relation:

$$b(x) \in B(x)[x \in A] \sim b(y) \in B(y)[y \in A]$$

Moreover, we write b for $(x)b(x)$. Actually, in order to have such expressions we should pass to the type theory with higher arity [NPS90], with the warning that what is called a type here, in [NPS90] is called

a set. By adding the so called function type, given $b(x) \in B(x)[x \in A]$ we have the abstraction, that is $(x)b \in (x \in A)B(x)$, the application, the β -conversion and the η -conversion, that is $(x)b(x) = b \in (x \in A)B(x)$.

In the previous calculi the types and corresponding terms coming from extensional Martin-Löf's type theory [Mar84] are those in the calculus for locally cartesian closed categories together with the disjoint sum types (but without the disjointness axiom). The extensional quotient types on any (equivalence) relations - hence not restricted to mono equivalence relations as we do here - and without effectiveness axiom, appeared in Nuprl [Con86].

Equivalent formulation of elimination rule for Indexed Sum types. Actually, from now on, we will refer to an equivalent formulation of the lex calculus where the elimination and conversion rules for the indexed sum type are replaced by the following rules:

$$\begin{array}{l}
\text{E}_1\text{-}\Sigma \quad \frac{d \in \Sigma_{x \in B} C(x)}{\pi_1^B(d) \in B} \quad \text{E}_2\text{-}\Sigma \quad \frac{d \in \Sigma_{x \in B} C(x)}{\pi_2^{C(\pi_1(d))}(d) \in C(\pi_1(d))} \\
\beta_1\text{C-}\Sigma \quad \frac{b \in B \quad c \in C(b)}{\pi_1^B(\langle b, c \rangle) = b \in B} \quad \beta_2\text{C-}\Sigma \quad \frac{b \in B \quad c \in C(b)}{\pi_2^{C(b)}(\langle b, c \rangle) = c \in C(b)} \\
\eta\text{C-}\Sigma \quad \frac{d \in \Sigma_{x \in B} C(x)}{\langle \pi_1^B(d), \pi_2^{C(\pi_1(d))}(d) \rangle = d \in \Sigma_{x \in B} C(x)}
\end{array}$$

The two existential quantifiers: strong and weak. One of the main point of Martin-Löf's type theory is the validity of the isomorphisms *propositions as types*. As a consequence the existential quantifier is put in correspondence with the indexed sum type as presented in the lex calculus. This means that we can define the two projections, as seen just before, with the consequence of having the existence property internalized in the calculus. This is stronger than the usual formalization of the intuitionistic existential quantifier, also present in topos theory, which corresponds to the weak indexed sum type according to the isomorphism *propositions as types*:

Weak Indexed Sum type

$$\begin{array}{l}
\frac{C(x) \text{ type } [x \in B]}{\Sigma_{x \in B}^w C(x) \text{ type}} \text{ w}\Sigma \quad \frac{b \in B \quad c \in C(b)}{\langle b, c \rangle \in \Sigma_{x \in B}^w C(x)} \text{ I-w}\Sigma \\
\\
\frac{\begin{array}{l} M \text{ type} \\ d \in \Sigma_{x \in B}^w C(x) \quad m(x, y) \in M [x \in B, y \in C(x)] \end{array}}{\text{split}_w(d, m) \in M} \text{ E-w}\Sigma \\
\\
\frac{\begin{array}{l} M \text{ type} \\ b \in B \quad c \in C(b) \quad m(x, y) \in M [x \in B, y \in C(x)] \end{array}}{\text{split}_w(\langle b, c \rangle, m) = m(b, c) \in M} \text{ C-w}\Sigma
\end{array}$$

where the difference with the indexed sum is that the type M must not depend on $\Sigma_{x \in B} C(x)$. This difference in the elimination rules gives rise to the fact that while in the presence of the product type the indexed sum type allows to prove the axiom of choice [Mar84]

$$((\Pi x \in A)(\Sigma y \in B) C(x, y)) \rightarrow ((\Pi f \in A \rightarrow B)(\Pi x \in A) C(x, f(x)))$$

this axiom does not follow from the weak one [Swa91, Swa92].

About the calculus of regular categories. In order to make a valid interpretation of the rules for the mono existential type in \mathcal{T}_{reg} it would be useful to note first that the mono existential type is actually mono, by using the elimination rule on the mono type $\text{Eq}(\exists_{x \in B} C(x) z, w)$ for $z \in \exists_{x \in B} C(x), w \in \exists_{x \in B} C(x)$.

Moreover, the rules of the mono existential type as formulated in \mathcal{T}_{reg} are equivalent to the following rules:

$$\begin{array}{c}
\frac{C(x) \text{ type } [x \in B] \quad y = z \in C(x) [x \in B, y \in C(x), z \in C(x)]}{\exists_{x \in B} C(x) \text{ type}} m\exists \\
\frac{b \in B \quad c \in C(b) \quad \exists_{x \in B} C(x) \text{ type}}{(b, c) \in \exists_{x \in B} C(x)} \text{I-}m\exists \quad \frac{e \in \exists_{x \in B} C(x) \quad d \in \exists_{x \in B} C(x)}{e = d \in \exists_{x \in B} C(x)} \text{eq-}m\exists \\
\frac{\begin{array}{c} M \text{ type} \quad y = z \in M [w \in \exists_{x \in B} C(x), y \in M, z \in M] \\ d \in \exists_{x \in B} C(x) \quad m(x, y) \in M [x \in B, y \in C(x)] \end{array}}{\text{Ex}_m(d, m) \in M(d)} \text{E-}m\exists
\end{array}$$

Of course the above rule are valid for the formulation of the mono existential type in \mathcal{T}_{reg} . To prove that from them we can derive the elimination rule on a mono dependent type $M(z) [z \in \exists_{x \in B} C(x)]$ with $m(x, y) \in M((x, y)) [x \in A, y \in C(x)]$ then apply the elimination rule E- $m\exists$ on $\Sigma_{z \in \exists_{x \in B} C(x)} M(z)$, which is mono since both $\exists_{x \in B} C(x)$ and $M(z)$ are mono. Then, use the second projection to define $\text{Ex}(z, m) \in M(z)$. This is well defined since $\pi_1(\text{Ex}_m(z, (x)(y)\langle(x, y), m(x, y)\rangle)) = z$ for $z \in \exists_{x \in B} C(x)$ being $\exists_{x \in B} C(x)$ mono.

However, from the elimination rule of the mono existential type we can derive the following stronger elimination on general types and corresponding β -conversion:

$$\begin{array}{c}
\frac{\begin{array}{c} M(z) \text{ type } [z \in \exists_{x \in B} C(x)] \quad m(x, y) \in M((x, y)) [x \in B, y \in C(x)] \\ d \in \exists_{x \in B} C(x) \quad m(x, y) = m(z, w) \in M((x, y)) [x \in B, z \in B, y \in C(x), w \in C(z)] \end{array}}{\text{Ex}_g(d, m) \in M(d)} \text{E-}g\exists \\
\frac{\begin{array}{c} M(z) \text{ type } [z \in \exists_{x \in B} C(x)] \quad m(x, y) \in M((x, y)) [x \in B, y \in C(x)] \\ a \in A \quad c \in C(b) \quad m(x, y) = m(z, w) \in M((x, y)) [x \in B, z \in B, y \in C(x), w \in C(z)] \end{array}}{\text{Ex}_g((b, c), m) = m(b, c) \in M((b, c))} \text{C-}g\exists
\end{array}$$

To prove that the elimination rule E- $g\exists$ is valid, we apply the elimination rule E- \exists of the mono existential type in \mathcal{T}_{reg} on the following mono type

$$\Sigma_{w \in M(z)} \exists_{x \in B} \exists_{y \in C(x)} \text{Eq}(M(z), w, m(x, y))$$

for $z \in \exists_{x \in B} C(x)$. Then, we use the first projection of the indexed sum type to define the elimination rule, i.e.

$$\text{Ex}_g(d, m) \equiv \pi_1(\text{Ex}(d, (x)(y)\langle m(x, y), (x, (y, \text{eq}))\rangle)) \in M(d)$$

The conversion rule C- $g\exists$ follows since

$$\pi_1(\text{Ex}((a, b), (x)(y)\langle m(x, y), (x, (y, \text{eq}))\rangle)) = \pi_1(\langle m(a, b), (a, (b, \text{eq}))\rangle) = m(a, b)$$

being $\Sigma_{w \in M(z)} \exists_{x \in B} \exists_{y \in C(x)} \text{Eq}(M(z), w, m(x, y))$ a mono type.

After noting this it is immediate to prove that from the existential type we can define

$$A/\top \equiv \exists_{x \in A} \top$$

Viceversa, from the quotient type A/\top we can define the existential type putting

$$\exists_{x \in B} C(x) \equiv (\Sigma_{x \in B} C(x))/\top$$

Finally, we note that the quotient of a kernel pair, that we would define as $A/\text{Eq}(B, f(x), f(y))$ for $f(x) \in B [x \in A]$, can be defined as

$$A/\text{Eq}(B, f(x), f(y)) \equiv \Sigma_{y \in B} (\exists_{x \in A} \text{Eq}(B, f(x), y))$$

In [AB01] there is an equivalent formulation of the calculus for regular categories with bracket types $[A]$ for a type A corresponding to our quotients on terminal type, i.e. $[A] \equiv A/\top$ or better $[A] \equiv \exists_{x \in A} \top$. The formulation of elimination rule for bracket types, which acts on non-dependent types, can be enforced

to act on dependent types and - once done this - the equality rule can be weakened to the equality rule only on canonical elements, as in our different formulations of rule for $\exists_{x \in A} \top$ or A/\top . The proof of this follows from what just said about the existential type, since the rules of bracket types make the rules $m\exists$ valid.

About the calculus for distributive categories and pretopoi. Note that in the calculi for distributive categories, locoi, pretopoi, Heyting pretopoi and arithmetic universes, the disjointness axiom is not derivable from the other rules. Indeed, we can obtain a model for each calculus that falsifies disjointness by using a domain with only one element (see [Smi88]), where the quotient type A/R is interpreted as A).

Mono types. We call *mono* every type for which we can prove

$$\frac{B(x) \text{ type } [x \in A]}{y = z \in B(x) [x \in A, y \in B(x), z \in B(x)]}$$

also called proof-irrelevant, for example in [Hof95]. These are dependent types that can be inhabited with at most one proof. This is the central concept to characterize the structure of subobjects of a category, like stable images of a regular category - captured by the mono existential types - or the right adjoints restricted to subobjects of an Heyting pretopos - captured by the forall types -. We can prove that the mono existential type and the forall type are mono respectively in \mathcal{T}_{reg} and in \mathcal{T}_{hptop} .

Remark on how to weaken the elimination rules on dependent types. In \mathcal{T}_{ptop} we can restrict the elimination rule of the quotient type to types not depending on the quotient type itself by adding an extra η -conversion rule. The same can be done for disjoint sum types, list types and natural numbers object type, as soon as they are together with the indexed sum type and the extensional equality type in order to make the η -conversion rule valid. We sketch the proof only in the case of the quotient type, since for the other types is analogous.

In \mathcal{T}_{ptop} the elimination and conversion rules of the quotient type are derivable, by using the indexed sum type, from the following restricted elimination rule of the quotient type for types not depending on A/R ,

E_s-Q

$$\frac{\begin{array}{c} M \text{ type} \\ d \in A/R \quad m(x) \in M [x \in A] \quad m(x) = m(y) \in M [x \in A, y \in A, d \in R(x, y)] \end{array}}{\mathbf{Q}_s(m, d) \in M}$$

together with the following two conversion rules, also derivable in \mathcal{T}_{ptop} : one is the β -conversion

β_s C-Q

$$\frac{\begin{array}{c} M \text{ type} \\ a \in A \quad m(x) \in M [x \in A] \quad m(x) = m(y) \in M [x \in A, y \in A, d \in R(x, y)] \end{array}}{\mathbf{Q}_s(m, [a]) = m(a) \in M}$$

and the other one is the η -conversion stating the uniqueness of \mathbf{Q}_s :

η_s C-Q

$$\frac{t(z) \in M [z \in A/R]}{\mathbf{Q}_s((x)t([x]), z) = t(z) \in M [z \in A/R]}$$

Indeed, given the judgements $l(x) \in L([x])[x \in C]$ and $l(x) = l(y) \in L([x]) [x \in A, y \in A, d \in R(x, y)]$, we use the \mathbf{E}_s -Q rule on $\langle [x], l(x) \rangle \in \Sigma_{z \in A/R} L(z) [x \in A]$. Then we define $\mathbf{Q}(l, p) \in L(p)$ for $p \in A/R$ as the second projection of the indexed sum type applied to $\mathbf{Q}_s((x)\langle [x], l(x) \rangle, p)$. It turns out to be well defined since by β_s and η_s conversion rules we can prove that

$$\pi_1(\mathbf{Q}_s((x)\langle [x], l(x) \rangle, z)) = z \in A/R [z \in A/R]$$

Similarly, in \mathcal{T}_{alex} the elimination and the conversion rules of the natural numbers type are derivable, by using the indexed sum type, from the following restricted elimination rule of the natural numbers type for types not depending on N

E_s-Nat

$$\frac{\begin{array}{c} L \text{ type} \\ n \in N \quad a \in L \quad l(y) \in L [y \in L] \end{array}}{\mathbf{Rec}_s(a, l, n) \in L}$$

together with the following three conversion rules, also derivable in \mathcal{T}_{alex} : two are the β -conversions $\beta_s \mathbf{C}_1\text{-Nat}$

$$\frac{L \text{ type} \quad a \in L \quad l(y) \in L [y \in L]}{\text{Rec}_s(a, l, 0) = a \in L}$$

$\beta_s \mathbf{C}_2\text{-Nat}$

$$\frac{L \text{ type} \quad n \in N \quad a \in L \quad l(y) \in L [y \in L]}{\text{Rec}_s(a, l, s(n)) = l(\text{Rec}_s(a, l, n)) \in L}$$

and the other one is the η -conversion stating the uniqueness of Rec_s :

$\eta_s \mathbf{C}\text{-Nat}$

$$\frac{a \in L \quad L \text{ type} \quad l(y) \in L [y \in L] \quad t(n) \in L [n \in N] \quad n \in N \quad t(0) = a \in L \quad t(s(n)) = l(t(n)) \in L}{\text{Rec}_s(a, l, n) = t(n) \in L}$$

Analogously, in \mathcal{T}_{loc} the elimination and conversion rules of the list type are derivable, by using the indexed sum type, from the following restricted elimination rule of the list type for types not depending on $List(C)$

$\mathbf{E}_s\text{-list}$

$$\frac{L \text{ type} \quad s \in List(C) \quad a \in L \quad l(z, x) \in L [z \in L, x \in C]}{\text{Rec}_s(a, l, s) \in L}$$

together with the following three conversion rules, also derivable in \mathcal{T}_{loc} : two are the β -conversions

$\beta_s \mathbf{C}_1\text{-list}$

$$\frac{L \text{ type} \quad a \in L \quad l(z, x) \in L [z \in L, x \in C]}{\text{Rec}_s(a, l, 0) = a \in L}$$

$\beta_s \mathbf{C}_2\text{-list}$

$$\frac{L \text{ type} \quad s \in List(C) \quad c \in C \quad a \in L \quad l(z, y) \in L [z \in L, y \in C]}{\text{Rec}_s(a, l, \text{cons}(s, c)) = l(\text{Rec}_s(a, l, s), c) \in L}$$

and the other one is the η -conversion stating the uniqueness of Rec_s

$\eta_s \mathbf{C}\text{-list}$

$$\frac{L \text{ type} \quad a \in L \quad l(z, y) \in L [z \in L, y \in C] \quad t(z) \in L [z \in List(C)] \quad s \in List(C) \quad t(\epsilon) = a \in L \quad t(\text{cons}(x, y)) = l(t(x), y) \in L [x \in List(C), y \in C]}{\text{Rec}_s(a, l, s) = t(s) \in L}$$

Finally, also in \mathcal{T}_{dis} the elimination and conversion rules of the disjoint sum type are derivable, by using the indexed sum type, from the following restricted elimination rule of the quotient type for types not depending on $A + B$

$\mathbf{E}_s\text{-+}$

$$\frac{A \text{ type} \quad w \in C + D \quad a_C(x) \in A [x \in C] \quad a_D(y) \in A [y \in D]}{\mathcal{D}_s(w, a_C, a_D) \in A}$$

and from the conversion rules:

$\mathbf{C}_{1s}\text{-+}$

$$\frac{A \text{ type} \quad c \in C \quad a_C(x) \in A [x \in C] \quad a_D(y) \in A [y \in D]}{\mathcal{D}_s(\text{inl}(c), a_C, a_D) = a_C(c) \in A}$$

C_{2s-+}

$$\frac{d \in D \quad a_C(x) \in A [x \in C] \quad a_D(y) \in A [y \in D] \quad \text{\textit{A type}}}{\mathcal{D}_s(\text{inr}(d), a_C, a_D) = a_D(d) \in A}$$

$\eta-+$

$$\frac{t \in A [z \in C + D]}{\mathcal{D}_s(z, (x)t(\text{inl}(x)), (y)t(\text{inr}(x))) = t(z) \in A}$$

Axiom of choice in the calculus for locally cartesian closed categories and topoi. Note that in the calculus for locally cartesian closed categories and topoi we can easily derive the following axiom of choice (see [Mar84])

$$((\Pi x \in A)(\Sigma y \in B) C(x, y)) \rightarrow ((\Pi f \in A \rightarrow B)(\Pi x \in A) C(x, f(x)))$$

where we recall that $A \rightarrow B \equiv (\Pi x \in A)B$.

However, note that this is not the usual propositional axiom of choice as in the internal language of topoi [LS86, MM92]. Indeed, in the formulation below we consider *predicates as dependent types* as in Martin-Löf's type theory. But in topoi, since the logic is described by the structure of subobjects, then we have that the isomorphism *predicates as mono dependent types* holds, since mono types correspond to monomorphism, as we will see in the next.

Hence, the propositional axiom of choice according to *predicates as mono dependent types* or *propositions as closed mono types* should be expressed in the type theory of an Heyting pretopos as

$$((\forall x \in A)(\exists y \in B) C(x, y)) \rightarrow ((\exists f \in A \rightarrow B)(\forall x \in A) C(x, f(x)))$$

supposing $C(x, y)$ a mono type and considering the existential type \exists as defined in the calculus for regular categories. This type in a pretopos, which is also a regular category, can be regained as

$$(1) \quad \exists y \in B C(y) \equiv (\Sigma y \in B C(y))/\top$$

A topos is also an Heyting pretopos. There the forall type is just a restriction of the dependent product type and the existential quantifier can be regained as follows:

$$\exists y \in B C(y) \equiv \Pi_{p \in \Omega} (\Pi_{y \in B} (C(y) \rightarrow \text{Eq}(\Omega, \{\top\}, p))) \rightarrow \text{Eq}(\Omega, \{\top\}, p)$$

and the quotient type on the terminal type can be obtained as $A/\top \equiv \exists x \in A \top$.

In conclusion, writing the existential quantifier as in (1), the propositional axiom of choice in a topos becomes

$$((\forall x \in A)((\Sigma y \in B C(x, y))/\top) \rightarrow ((\Sigma f \in A \rightarrow B)(\forall x \in A) C(x, f(x)))/\top)$$

We recall that this axiom can not be derived generally in a topos. Indeed, following Diaconescu's proof [Dia75] (see also [LS86, MM92]) this axiom makes the logic of a topos classical. Here, we can realize why this axiom can not be derived in a generic topos, because of the impossibility of accessing to a proof of $\Sigma y \in B C(x, y)$ from $(\Sigma y \in B C(x, y))/\top$, unless we have a choice operator from the quotient A/\top to A .

Instead, in a topos (and also in a Heyting pretopos) we can derive the axiom of unique choice written as

$$((\forall x \in A) (((\Sigma y \in B C(x, y))/\top) \wedge \forall y \in B \forall z \in B C(x, y) \wedge C(x, z) \rightarrow \text{Eq}(B, y, z)) \rightarrow ((\Sigma f \in A \rightarrow B)(\forall x \in A) C(x, f(x)))/\top)$$

where $A \wedge B \equiv A \times B$ with A, B mono types. Indeed, in this case $\Sigma y \in B C(x, y) [x \in A]$ is a mono type and then the above type becomes isomorphic to its copy without the quotient

$$(\Sigma y \in B C(x, y))/\top \simeq \Sigma y \in B C(x, y)$$

and then we can prove the axiom of unique choice as we prove the axiom of choice in Martin-Löf's type theory.

Omega type of topos calculus seen as a quotient type. The novelty of the typed calculus for topoi with respect to already known internal languages of topoi, for example in [LS86], is the Omega type corresponding to the subobject classifier that encodes propositions represented as closed mono types. Hence, the impredicativity of a topos is restricted to mono types but the Omega type is not necessarily itself a mono type.

In the Omega type mono types are encoded up to equiprovability. To make this clearer we can see how the Omega type is actually the quotient of an intensional classifier over the equiprovability relation. Before we explain why in the formulation of the Omega type an elimination rule is not present and where the β -conversion comes from, looking at the following alternative formulation of the rules for the Omega type:

$$\begin{array}{c}
\textbf{Alternative formulation of the Omega type} \\
\hline
\frac{}{\Omega \text{ type}} \mathbf{F} \\
\frac{B \text{ type} \quad y = z \in B \ [y \in B, z \in B]}{\{B\} \in \Omega} \mathbf{I} \quad \frac{C \text{ type} \quad y = z \in C \ [y \in C, z \in C] \quad f \in B \leftrightarrow C}{\{B\} = \{C\} \in \Omega} \mathbf{eq} \\
\frac{q \in \Omega}{T(q) \text{ type}} \mathbf{E} \quad \frac{q \in \Omega \quad c \in T(q) \quad d \in T(q)}{c = d \in T(q)} \mathbf{eq-E} \\
\frac{B \text{ type} \quad y = z \in B \ [y \in B, z \in B]}{\langle r_B, r_B^{-1} \rangle \in T(\{B\}) \leftrightarrow B} \beta\text{-C} \quad \frac{q \in \Omega}{\{T(q)\} = q \in \Omega} [\eta\text{-C}]
\end{array}$$

In the above alternative formulation of the Omega type its elimination rule is explicit. However, adding the above rules to \mathcal{T}_{lcc} for every $q \in \Omega$ we can derive a proof term of

$$T(q) \leftrightarrow \mathbf{Eq}(\Omega, q, \{\top\})$$

Hence, we can put

$$T(q) \equiv \mathbf{Eq}(\Omega, q, \{\top\})$$

and simply keep only the corresponding β and η conversion rules after the above substitution to obtain the rules of \mathcal{T}_{top} .

We can also weaken the β -conversion of the Omega type in \mathcal{T}_{top} to the following

$$\beta_r \mathbf{C}\text{-}\Omega) \quad \frac{B \text{ type} \quad y = z \in B \ [y \in B, z \in B]}{r_B \in \mathbf{Eq}(\Omega, \{B\}, \{\top\}) \rightarrow B}$$

since we can always prove that

$$r_B^{-1} = \lambda z. \mathbf{eq} \in B \rightarrow \mathbf{Eq}(\Omega, \{B\}, \{\top\})$$

We conclude showing that the rules of \mathcal{T}_{top} can be derived inside an extension of \mathcal{T}_{lcc} with extensional effective quotients restricted to mono equivalence relations, as in the type theory of pretopoi \mathcal{T}_{ptop} , and

with an intensional Omega type encoding mono types as follows:

The intensional Omega type	
$\frac{}{\Omega^i \text{ type}} \mathbf{F}$	
$\frac{B \text{ type } y = z \in B [y \in B, z \in B]}{c(B) \in \Omega^i} \mathbf{I}$	$\frac{\begin{array}{l} B \text{ type } y = z \in B [y \in B, z \in B] \\ C \text{ type } y = z \in C [y \in C, z \in C] \\ B = C \end{array}}{c(B) = c(C) \in \Omega^i} \mathbf{eq}$
$\frac{p \in \Omega^i}{D(p) \text{ type}} \mathbf{E}$	$\frac{p \in \Omega^i \quad c \in D(p) \quad d \in D(p)}{c = d \in D(p)} \mathbf{eq-E}$
$\frac{B \text{ type } y = z \in B [y \in B, z \in B]}{D(c(B)) = B} \beta\text{-C}$	$\frac{p \in \Omega^i}{c(D(p)) = p \in \Omega^i} \eta\text{-C}$

In this extension we can prove that

$$\Omega \equiv \Omega^i / \leftrightarrow$$

Then we can see how the equality rule is the usual rule of equality for canonical terms of a quotient type and that the β -conversion rule of the Omega type in \mathcal{T}_{ptop} corresponds to effectiveness of the Omega type seen as a quotient.

Intensional and extensional type theories. We recall that the extensional type theory is characterized by the extensional equality type

$$\frac{C \text{ type } c \in C \quad d \in C}{\mathbf{Eq}(C, c, d) \text{ type}} \mathbf{Eq} \quad \frac{c \in C}{\mathbf{eq}_C(c) \in \mathbf{Eq}(C, c, c)} \mathbf{I-Eq}$$

$$\frac{p \in \mathbf{Eq}(C, c, d)}{c = d \in C} \mathbf{E-Eq} \quad \frac{p \in \mathbf{Eq}(C, c, d)}{p = \mathbf{eq}_C(c) \in \mathbf{Eq}(C, c, d)} \mathbf{C-Eq}$$

Instead, the intensional type theory [NPS90] is characterized by the fact that all the type constructors have only β -conversions and the equality type is intensional:

Intensional equality type	
$\frac{A \text{ type } a \in A \quad b \in A}{\mathbf{ld}(A, a, b) \text{ type}} \mathbf{ld}$	$\frac{a \in A}{\mathbf{id}(a) \in \mathbf{ld}(A, a, a)} \mathbf{I-Id}$
$d \in \mathbf{ld}(A, a, b) \quad C(x, y, z) [x \in A, y \in A, z \in \mathbf{ld}(A, x, y)]$	$c(x) \in C(x, x, \mathbf{id}(x)) [x : A]$
$\frac{}{\mathbf{idpeel}(d, c) \in C(a, b, d)} \mathbf{E-Id}$	
$C(x, y, z) [x \in A, y \in A, z \in \mathbf{ld}(A, x, y)]$	
$\frac{a \in A \quad c(x) \in C(x, x, \mathbf{id}(x)) [x : A]}{\mathbf{idpeel}(\mathbf{id}(a), c) = c(a) \in C(a, a, \mathbf{id}(a))} \mathbf{C-Id}$	

A big difference between intensional and extensional versions of type theory is that decidability of definitional equality $a = b \in A$ follows in the intensional version, while this is no longer the case for the extensional version (see for example [Hof95] for discussions about this). Moreover, in general η -equalities

for the constructors are not valid in an intensional framework, and they are generally derivable if the extensional equality is added to the intensional version.

However, even in an intensional framework with fragments of Martin-Löf's type theory Diaconescu's proof still can be reproduced to see the incompatibility of extensional constructors, like extensional powersets or effective quotient types, with constructivity of Martin-Löf's type theory and its isomorphism *propositions as types* (see [MV99, Mai99a]). In other terms the restriction to mono types in the formulation of Omega type and effective quotient types is crucial to avoid classical logic.

Why categories correspond to extensional type theories. In our correspondence between categories and dependent typed languages the internal dependent type theories we show as internal languages of the considered categories are all extensional. The reason is that in the syntactic category with closed types as objects and terms as morphisms, the equality between morphisms is taken to be the definitional equality. Hence, equalizers correspond exactly to extensional equality types according to our categorical semantics. As far as we know, no alternative choice of equality between morphisms of the syntactic categories with closed types and terms can be put to establish a link between categories and (possibly) dependent typed calculi.

4 The modular correspondence categorical property/type constructor

Here, we describe the correspondence between universal categorical properties and type constructors. It is worth to recall that the correspondence between typed calculi and categories we derive is not only based on validity and completeness theorems but on the stronger link that the dependent type theories provide the internal languages of the corresponding categories in which they are modelled.

Categorical properties	Type-theoretic constructors
Finite limits	terminal type extensional equality type indexed sum types
+	+
stable finite disjoint sums	disjoint sum types falsum type disjointness axiom
stable images	mono existential type restricted to mono types
stable quotients of kernel pairs	extensional quotient types on the terminal type
stable effective quotients of mono equivalence relations	extensional quotient types effectiveness axiom
parameterized natural numbers object	natural numbers type
parameterized list objects	list types
right adjoint to pullback functor	extensional dependent product types
right adjoint to pullback functor restricted to subobjects	dependent product types restricted to mono types
subobject classifier	omega type

Other calculi for other categories. Since the above correspondence is modular on lex categories, then we can deduce the dependent type theory of other lex categories enjoying a combination of the above categorical properties by combining the corresponding type constructors. For example, the type theory of locally cartesian closed pretopoi is the calculus of pretopoi with the product type, or the type theory of locally cartesian closed categories with stable finite coproducts is the calculus of locally cartesian closed categories with the falsum type and the disjoint sum type.

5 The syntactic categories out of the type theories

The aim of this section is to show how every typed calculus proposed in the previous sections gives rise to a syntactic category with the categorical properties it intends to capture, modularly as in the table of section 4. We start showing how to build a lex category out of the calculus \mathcal{T}_{lex} to end with the syntactic topos out of the calculus \mathcal{T}_{top} . These categories are of course useful to prove a completeness theorem between a calculus and the class of categories we mean to describe with it.

It is worth noting that contrary to the syntactic topos in [LS86], using dependent types we build a *proof-relevant* topos. Indeed, here morphisms are terms instead of functional relations there. The same can be said for our syntactic lex and regular categories with respect to those built with the calculi in terms of many-sorted logic already known in the literature.

From now on we refer to $\mathcal{C}_{\mathcal{T}}$ as the syntactic category built from the dependent typed calculus \mathcal{T} .

We define the syntactic category $\mathcal{C}_{\mathcal{T}}$ as follows:

Def. 5.1 The objects of $\mathcal{C}_{\mathcal{T}}$ are the closed types of \mathcal{T} , $A, B, C \dots$ and the morphisms between two types, A and B , are the expressions $(x)b(x)$ (see [NPS90]) corresponding to

$$b(x) \in B [x \in A]$$

where the type B does not depend on A . The composition in $\mathcal{C}_{\mathcal{T}}$ is defined by substitution, that is given $(x)b(x) \in \mathcal{C}_{\mathcal{T}}(A, B)$ and $(y)c(y) \in \mathcal{C}_{\mathcal{T}}(B, C)$ their composition is $(x)c(b(x))$. We state that $(x)b(x) \in P(A, B)$ and $(x)b'(x) \in P(A, B)$ are equal iff we can derive

$$b(x) = b'(x) \in B[x \in A]$$

The identity is $(x)x \in P(A, A)$ obtained by $x \in A[x \in A]$.

As already said, the choice of definitional equality as equality of morphisms is crucial to make a correspondence between categories and extensional dependent type theories.

5.1 Lex category

In this section we are going to prove that

Proposition 5.2 *The category $\mathcal{C}_{\mathcal{T}_{lex}}$ is finitely complete.*

The *terminal object* is \top and from any object A the morphism towards \top is

$$(x)\star \in \mathcal{C}_{\mathcal{T}_{lex}}(A, \top)$$

which is unique by the conversion rule for \top .

Given $c \in \mathcal{C}_{\mathcal{T}_{lex}}(A, C)$ and $d \in \mathcal{C}_{\mathcal{T}_{lex}}(B, C)$ the *pullback* is given by

$$\Sigma_{x \in A} \Sigma_{y \in B} \mathbf{Eq}(C, c(x), d(y))$$

where the first projection to A is

$$(z)\pi_1^A(z) \in \mathcal{C}_{\mathcal{T}_{lex}}(\Sigma_{x \in A} \Sigma_{y \in B} \mathbf{Eq}(C, c(x), d(y)), A)$$

and the second projection to B is

$$(z)\pi_1^B(\pi_2^A(z)) \in \mathcal{C}_{\mathcal{T}_{lex}}(\Sigma_{x \in A} \Sigma_{y \in B} \mathbf{Eq}(C, c(x), d(y)), B)$$

In the following, we will often write $a =_A b$ to mean $\mathbf{Eq}(A, a, b)$ and \mathbf{eq}_C , instead of $\mathbf{eq}_C(c)$.

5.2 The disjoint coproduct

In $\mathcal{C}_{\mathcal{T}_{dis}}$ the *coproduct* of A and B is defined by $A + B$, where the injections are

$$(x) \text{ inl}(x) \in \mathcal{C}_{\mathcal{T}_{dis}}(A, A + B) \quad \text{and} \quad (y) \text{ inr}(y) \in \mathcal{C}_{\mathcal{T}_{dis}}(B, A + B)$$

Given $c \in \mathcal{C}_{\mathcal{T}_{dis}}(A, C)$ and $d \in \mathcal{C}_{\mathcal{T}_{dis}}(B, C)$ the mediating morphism $c \oplus d$ from $A + B$ to C is $(w) \mathcal{D}(w, c, d)$. Coproducts are disjoint by the rule of disjointness. Moreover, coproducts are stable under pullback. For this purpose, we prove that

Lemma 5.3 $A + B$ is isomorphic in $\mathcal{C}_{\mathcal{T}_{dis}}$ to

$$\Sigma_{w \in A+B} (\Sigma_{x \in A} \text{inl}(x) =_A w) + (\Sigma_{y \in B} \text{inr}(y) =_B w)$$

and in particular $(z) \pi_1(z) \in \mathcal{C}_{\mathcal{T}_{dis}}(\Sigma_{w \in A+B} \tilde{A}(w) + \tilde{B}(w), A + B)$ has got an inverse

$$\delta \in \mathcal{C}_{\mathcal{T}_{dis}}(A + B, \Sigma_{w \in A+B} \tilde{A}(w) + \tilde{B}(w))$$

where $\tilde{A}(w) \equiv \Sigma_{x \in A} \text{inl}(x) =_{A+B} w$ $\tilde{B}(w) \equiv \Sigma_{y \in B} \text{inr}(y) =_{A+B} w$.

and hence we can prove

Proposition 5.4 In $\mathcal{C}_{\mathcal{T}_{dis}}$ coproducts are stable under pullbacks.

Proof.

Given the following pullbacks

$$\begin{array}{ccc} P_1 & \xrightarrow{\pi_2^1} & A \\ \pi_1^1 \downarrow & & \downarrow a \\ D & \xrightarrow{m} & C \end{array} \quad \begin{array}{ccc} P_2 & \xrightarrow{\pi_2^2} & B \\ \pi_1^2 \downarrow & & \downarrow b \\ D & \xrightarrow{m} & C \end{array} \quad \begin{array}{ccc} P & \xrightarrow{\pi_2^P} & A + B \\ \pi_1^P \downarrow & & \downarrow a \oplus b \\ D & \xrightarrow{m} & C \end{array}$$

we have to show that in $\mathcal{C}_{\mathcal{T}_{dis}}/D$

$$\pi_1^1 \oplus \pi_1^2 \simeq \pi_1^P$$

For this purpose we define

$$\gamma : P_1 + P_2 \rightarrow P$$

as $\gamma \equiv (w) \mathcal{D}(w, d_1, d_2)$ where d_1 corresponds to

$$\langle \pi_1(w_1), \langle \text{inl}(\pi_1(\pi_2(w_1))), \text{eq}_C \rangle \rangle \in P [w_1 \in P_1]$$

and d_2 corresponds to

$$\langle \pi_1(w_2), \langle \text{inr}(\pi_1(\pi_2(w_2))), \text{eq}_C \rangle \rangle \in P [w_2 \in P_2]$$

We can notice that $\pi_1^P \cdot \gamma = \pi_1^1 \oplus \pi_1^2$ and that $\pi_2^P \cdot \gamma = (\text{inl} \cdot \pi_2^1) \oplus (\text{inr} \cdot \pi_2^2)$. Moreover, we want to define

$$\gamma^{-1} : P \rightarrow P_1 + P_2$$

First of all, we consider that, given $w \in P$, we get $\pi_1(\pi_2(w)) \in A + B$, hence, by δ defined in the above lemma we deduce

$$\pi_2(\delta(\pi_1(\pi_2(w)))) \in \tilde{A}(\pi_1(\pi_2(w))) + \tilde{B}(\pi_1(\pi_2(w)))$$

Now, we use the elimination rule with respect to $\tilde{A}(\pi_1(\pi_2(w))) + \tilde{B}(\pi_1(\pi_2(w)))$ and we define

$$\gamma^{-1} \equiv (w) \mathcal{D}(\pi_2(\delta(\pi_1(\pi_2(w))))), d'_1, d'_2)$$

where d'_1 corresponds to

$$\text{inl}(\langle \pi_1(w), \langle \pi_1(x'), \text{eq}_C \rangle \rangle) \in P_1 + P_2 [w \in P, x' \in \tilde{A}(\pi_1(\pi_2(w)))]$$

Indeed, from $w \in P$ and $x' \in \tilde{A}(\pi_1(\pi_2(w)))$ we get

$$m(\pi_1(w)) = (a \oplus b)(\pi_1(\pi_2(w))) \text{ and } \pi_1(\pi_2(w)) = \text{inl}(\pi_1(x'))$$

therefore $m(\pi_1(w)) = a(\pi_1(x'))$. In an analogous way, we define d'_2 as

$$\text{inr}(\langle \pi_1(w), \langle \pi_1(y'), \text{eq}_C \rangle \rangle) \in P_1 + P_2 [w \in P, y' \in \tilde{B}(\pi_1(\pi_2(w)))]$$

We can prove that γ^{-1} is the inverse morphism of γ by the elimination rule of the disjoint sum type.

■

5.3 The natural numbers object

The natural numbers object in $\mathcal{C}_{\mathcal{T}_{alex}}$ is the closed type N . Given a closed type Y the zero map is

$$(x)\langle x, 0 \rangle \in \mathcal{C}_{\mathcal{T}_{alex}}(Y, Y \times N)$$

and the successor map corresponds to $s(n) [n \in N]$. We put $id \times s \equiv (w)\langle \pi_2(w), s(\pi_2(w)) \rangle$. Given the morphisms $f \in \mathcal{C}_{\mathcal{T}_{alex}}(B, Y)$ and $g \in \mathcal{C}_{\mathcal{T}_{alex}}(Y, Y)$, we can prove that there exists an unique morphism $t \in \mathcal{C}_{\mathcal{T}_{alex}}(B \times N, Y)$ such that the following diagram commutes in all its parts:

$$\begin{array}{ccccc} B & \xrightarrow{\langle id, 0 \rangle} & B \times N & \xrightarrow{id \times s} & B \times N \\ & \searrow f & \downarrow \langle \pi_1, t \rangle & & \downarrow t \\ & & Y & \xrightarrow{g} & Y \end{array}$$

By hypothesis we get

$$f(y) \in Y [y \in B] \quad \text{and} \quad g(\langle y, z \rangle) \in Y [y \in B, z \in N]$$

By the elimination rule of the natural numbers type we derive

$$\text{Rec}_s(f(y), g, z) \in Y [y \in B, z \in N]$$

So, we put $t \equiv (w)\text{Rec}_s(f(\pi_1(w)), g, \pi_2(w))$, which is the required morphism to make the diagram commute by the conversion rules for the natural numbers type.

5.4 The list object

The syntactic category $\mathcal{C}_{\mathcal{T}_{loc}}$ is equipped with list-objects. The empty map is

$$(x)\epsilon \in \mathcal{C}_{\mathcal{T}_{loc}}(B, \text{List}(A))$$

and the list-constructor map is $(z)\text{cons}(\pi_1(z), \pi_2(z)) \in \mathcal{C}_{\mathcal{T}_{loc}}(\text{List}(A) \times A, \text{List}(A))$. Then, given a closed type Y and the morphisms $f \in \mathcal{C}_{\mathcal{T}_{loc}}(B, Y)$ and $g \in \mathcal{C}_{\mathcal{T}_{loc}}(Y \times A, Y)$ we can prove that there exists a unique morphism $t \in \mathcal{C}_{\mathcal{T}_{loc}}(B \times \text{List}(A), Y)$ such that the following diagram commutes in all its parts:

$$\begin{array}{ccccc} B & \xrightarrow{id \times \langle (x)\epsilon \rangle} & B \times \text{List}(A) & \xleftarrow{id \times \text{cons}} & B \times (\text{List}(A) \times A) \\ & \searrow f & \downarrow t & & \downarrow (t \times id) \cdot \sigma \\ & & Y & \xleftarrow{g} & Y \times A \end{array}$$

Indeed by the elimination rule of the list type we can define

$$t \equiv (z)\text{Rec}_s(f(\pi_1(z)), g, \pi_2(z)) \in \mathcal{C}_{\mathcal{T}_{loc}}(B \times \text{List}(A), Y)$$

which is the unique map making the diagrams commute by β_s and η_s C-list conversion rules.

5.5 The image

In $\mathcal{C}_{\mathcal{T}_{reg}}$ given a morphism $f \in \mathcal{C}_{\mathcal{T}_{reg}}(A, B)$, the image of f is

$$\mathcal{I}mf \equiv \pi_1 \in \mathcal{C}_{\mathcal{T}_{reg}}(\Sigma_{y \in B} \exists x \in A \mathbf{Eq}(B, f(x), y), B)$$

and for short we put $I(f) \equiv \Sigma_{y \in B} \exists x \in A \mathbf{Eq}(B, f(x), y)$. Indeed, we can define

$$q(z) \equiv \langle f(x), (x, \text{eq}) \rangle \in I(f) [z \in A]$$

and $f = \mathcal{I}mf \cdot q$.

Moreover, given another factorization $f = i \cdot p$ with $i \in \mathcal{P}_{\mathcal{T}_{reg}}(D, B)$ mono, then we can define for $z \in \Sigma_{y \in B} \exists x \in A \mathbf{Eq}(B, f(x), y)$

$$t(z) \equiv \mathbf{Ex}_g(\pi_2(z), (x)(y)p(x)) \in D$$

This is well defined since if $f(x) = \pi_1(z) = f(x')$ then $i(p(x)) = f(x) = f(x') = i(p(x'))$ and since i is mono we conclude $p(x) = p(x')$. Finally, we get that $i \cdot t = \pi_1$ by elimination rule of the mono existential type on the equality type

$$i(\mathbf{Ex}_g(w, (x)(y)p(x))) =_B \pi_1(z) \times \pi_2(z) =_{\exists x \in A \mathbf{Eq}(B, f(x), \pi_1(z))} w$$

since for $a \in A, b \in B$ such that $f(a) = b$ we have that $i(t(\langle b, (a, \text{eq}) \rangle)) = i(p(a)) = f(a) = b$.

Proposition 5.5 *In $\mathcal{C}_{\mathcal{T}_{reg}}$ the image of any $f \in \mathcal{C}_{\mathcal{T}_{reg}}(A, B)$*

$$\pi_1 \in \mathcal{C}_{\mathcal{T}_{reg}}(\Sigma_{y \in B} \exists x \in A \mathbf{Eq}(B, f(x), y), B)$$

is stable under pullbacks.

Proof. Given $h \in \mathcal{C}_{\mathcal{T}_{reg}}(C, B)$ we need to show that the image of $\pi_1^{D \times A}$ is $\pi_1^{D \times I(f)}$ in the following pullback squares

$$\begin{array}{ccc} P & \xrightarrow{\pi_2^{D \times A}} & A \\ \pi_1^{D \times A} \downarrow & & \downarrow f \\ D & \xrightarrow{h} & B \end{array} \quad \begin{array}{ccc} Q & \xrightarrow{\pi_2^{D \times I(f)}} & \Sigma_{y \in B} \exists x \in A \mathbf{Eq}(B, f(x), y) \\ \pi_1^{D \times I(f)} \downarrow & & \downarrow \mathcal{I}mf \\ D & \xrightarrow{h} & B \end{array}$$

where $P \equiv \Sigma_{y \in D} \Sigma_{x \in A} \mathbf{Eq}(C, h(y), f(x))$ and $Q \equiv \Sigma_{y \in D} \Sigma_{x \in I(f)} \mathbf{Eq}(C, \mathcal{I}mf(x), h(y))$.

It is sufficient to prove that there is an isomorphism between Q and

$$\Sigma_{y \in D} \exists w \in P \mathbf{Eq}(B, \pi_1^{D \times A}(w), y)$$

such that $\mathcal{I}m \pi_1^{D \times A}$ is isomorphic to $\pi_1^{D \times I(f)}$ in $\mathcal{C}_{\mathcal{T}_{reg}}/D$.

We define

$$\delta(z) \in Q [z \in \Sigma_{y \in D} \exists w \in P \mathbf{Eq}(B, \pi_1^{D \times A}(w), y)]$$

as follows

$$\delta(z) \equiv \langle \pi_1(z), \mathbf{Ex}_g(\pi_2(z), (w)(w')(\langle f(\pi_1(\pi_2(w))), (\pi_1(\pi_2(w)), \text{eq}), \text{eq}) \rangle) \rangle$$

Viceversa, we define

$$\delta^{-1}(w) \in \Sigma_{y \in D} \exists w \in P \mathbf{Eq}(B, \pi_1^{D \times A}(w), y) [w \in Q]$$

as follows

$$\delta^{-1}(w) \equiv \langle \pi_1(w), \mathbf{Ex}_g(\pi_2(\pi_1(\pi_2(w))), (x)(y)(\langle \pi_1(w), \langle x, \text{eq} \rangle, \text{eq} \rangle) \rangle$$

We can prove that δ and δ^{-1} are well defined and that they are inverse each other by the elimination rule of the existential type and the fact that it is mono.

■

5.6 The quotient of an equivalence relation

In $\mathcal{C}_{\mathcal{T}_{ptop}}$ given an equivalence relation

$$R \xrightarrow{g} A \times A$$

we consider the following mono type:

$$\mathcal{R}(x, x') \equiv \Sigma_{y \in R} g(y) =_{A \times A} \langle x, x' \rangle [x \in A, x' \in A]$$

It is easy to check that the categorical definition of equivalence relation implies that $\mathcal{R}(x, x')[x \in A, x' \in A]$ is an equivalence relation from the type-theoretical point of view. Let A/\mathcal{R} be the quotient with respect to $\mathcal{R}(x, x')[x \in A, x' \in A]$. We can prove that $(z)[z] \in \mathcal{C}_{\mathcal{T}_{ptop}}(A, A/\mathcal{R})$ is the coequalizer of $\pi_1 \cdot g \in \mathcal{C}_{\mathcal{T}_{ptop}}(R, A)$ and $\pi_2 \cdot g \in \mathcal{C}_{\mathcal{T}_{ptop}}(R, A)$ by the elimination and conversion rules of the quotient type. The uniqueness property of the coequalizer follows from the η_s C-quotient rule.

In $\mathcal{C}_{\mathcal{T}_{ptop}}$ the categorical equivalence relations are effective, that is $\pi_1 \cdot g$ and $\pi_2 \cdot g$ are projections (or kernel pair) for the pullback of $(z)[z]$ along itself. The existence of a morphism towards the pullback vertex is guaranteed by the effectiveness axiom and its uniqueness follows from the fact that equivalence relations are monic.

Moreover, we prove stability of quotients for equivalence relations.

Proposition 5.6 *In $\mathcal{C}_{\mathcal{T}_{ptop}}$ equivalence relations are stable effective.*

Proof.

In the following we write $\pi_i^{A \times D}$ for the i 'th projection from the vertex of the pullback of suitable arrows $A \rightarrow \cdot \leftarrow D$. We will omit to label the projections when their domains and codomains are clear from the context.

Given $m \in \mathcal{C}_{\mathcal{T}_{ptop}}(D, A/\mathcal{R})$ let us consider the following pullbacks:

$$\begin{array}{ccc} P & \xrightarrow{\pi_2^{D \times A}} & A \\ \pi_1^{D \times A} \downarrow & & \downarrow (z)[z] \\ D & \xrightarrow{m} & A/\mathcal{R} \end{array} \quad \begin{array}{ccc} Q & \xrightarrow{\pi_2^{D \times R}} & R \\ \pi_1^{D \times R} \downarrow & \begin{array}{c} \pi_1 \cdot g \downarrow \\ \pi_2 \cdot g \downarrow \end{array} & \downarrow (z)[z] \\ D & \xrightarrow{m} & A/\mathcal{R} \end{array}$$

where

$$P \equiv \Sigma_{w \in D} \Sigma_{x \in A} \text{Eq}(A/\mathcal{R}, m(w), [x]) \quad \text{and} \quad Q \equiv \Sigma_{w \in D} \Sigma_{y \in R} \text{Eq}(A/\mathcal{R}, m(w), [(\pi_1 \cdot g)(y)])$$

Moreover, let us consider these two pullbacks:

$$\begin{array}{ccc} Q & \xrightarrow{\pi_2^{D \times R}} & R \\ (\pi_2^{D \times A})^*(\pi_1 \cdot g) \downarrow & & \downarrow \pi_1 \cdot g \\ P & \xrightarrow{\pi_2^{D \times A}} & A \end{array} \quad \begin{array}{ccc} Q & \xrightarrow{\pi_2^{D \times R}} & R \\ (\pi_2^{D \times A})^*(\pi_2 \cdot g) \downarrow & & \downarrow \pi_2 \cdot g \\ P & \xrightarrow{\pi_2^{D \times A}} & A \end{array}$$

where

$$(\pi_2^{D \times A})^*(\pi_1 \cdot g) \equiv (w) \langle \pi_1(w), \langle \pi_1(g(\pi_1(\pi_2(w)))) \rangle, \text{eq} \rangle$$

and

$$(\pi_2^{D \times A})^*(\pi_2 \cdot g) \equiv (w) \langle \pi_1(w), \langle \pi_2(g(\pi_1(\pi_2(w)))) \rangle, \text{eq} \rangle$$

We must show that in $P/\mathcal{C}_{\mathcal{T}_{ptop}}$

$$\pi_1^{D \times A} \simeq \text{coeq}((\pi_2^A)^*(\pi_1 \cdot g), (\pi_2^A)^*(\pi_2 \cdot g))$$

We recall that the objects of the category $P/\mathcal{C}_{\mathcal{T}_{ptop}}$ are the morphisms $b : P \rightarrow B$ of \mathcal{C} , and the morphisms of $P/\mathcal{C}_{\mathcal{T}_{ptop}}$ from $b : P \rightarrow B$ to $b' : P \rightarrow B'$ are the morphisms $t : B \rightarrow B'$ of \mathcal{C} such that $t \cdot b = b'$. We can observe that the pullback given by the effectiveness

$$\begin{array}{ccc}
 R & \xrightarrow{\pi_2 \cdot g} & A \\
 \pi_1 \cdot g \downarrow & & \downarrow (z)[z] \\
 A & \xrightarrow{(z)[z]} & A/\mathcal{R} \\
 & \nearrow m & \\
 D & &
 \end{array}$$

can be completed in a cube of pullbacks and hence

$$\begin{array}{ccc}
 Q & \xrightarrow{(\pi_2^A)^*(\pi_2 \cdot g)} & P \\
 (\pi_2^A)^*(\pi_1 \cdot g) \downarrow & & \downarrow \pi_1^{D \times A} \\
 P & \xrightarrow{\pi_1^{D \times A}} & D
 \end{array}$$

is a pullback. Therefore $\langle \pi_2^*(\pi_1 \cdot g), \pi_2^*(\pi_2 \cdot g) \rangle$ is an equivalence relation as kernel pair of $\pi_1^{D \times A}$. Then let us consider the coequalizer of $\pi_2^*(\pi_1 \cdot g)$ and $\pi_2^*(\pi_2 \cdot g)$

$$[-]_P : P \rightarrow P/m^*(\mathcal{R})$$

where $P/m^*(\mathcal{R})$ is the quotient type with respect to the equivalence relation $\langle \pi_2^*(\pi_1 \cdot g), \pi_2^*(\pi_2 \cdot g) \rangle$. Since $\pi_1^{D \times A} \cdot \pi_2^*(\pi_1 \cdot g) = \pi_1^{D \times A} \cdot \pi_2^*(\pi_2 \cdot g)$ there exists a map

$$Q^P : P/m^*(\mathcal{R}) \rightarrow D$$

such that $Q^P \cdot [-]_P = \pi_1^{D \times A}$.

Now we want to prove that Q^P is an isomorphism in $P/\mathcal{C}_{\mathcal{T}_{ptop}}$.

In order to define the inverse of Q^P we need the following lemma:

Lemma 5.7 *For every equivalence relation R on A we prove that A/R is isomorphic to*

$$\Sigma_{z \in A/R} (\Sigma_{x \in A} ([x] =_{A/R} z)) / \top$$

Proof.

We define in \mathcal{T}_{ptop}

$$\phi : A/R \rightarrow \Sigma_{z \in A/R} (\Sigma_{x \in A} ([x] =_{A/R} z)) / \top$$

as $\phi(z) \equiv \langle z, \mathbf{Q}(z, (x)[\langle x, \text{eq} \rangle]) \rangle$ for $z \in A/R$ and

$$\phi^{-1} : \Sigma_{z \in A/R} (\Sigma_{x \in A} ([x] =_{A/R} z)) / \top \rightarrow A/R$$

as $\phi^{-1}(z') \equiv \pi_1(z')$ for $z' \in \Sigma_{z \in A/R} \Sigma_{x \in A} ([x] =_{A/R} z) / \top$. It is immediate to see $\phi^{-1} \cdot \phi = id$ and $\phi \cdot \phi^{-1} = id$ follows from the fact that $\Sigma_{x \in A} ([x] =_{A/R} z) / \top$ is a mono type for every fixed $z \in A/R$.

■

Now we go back to prove that Q^P is an isomorphism by finding its inverse. By means of ϕ defined in the above lemma we define: for $d \in D$

$$Q^{P^{-1}}(d) \equiv \mathbf{Q}(\pi_2(\phi(m(d))), (w)[\langle d, \langle \pi_1 w, \text{eq} \rangle \rangle])$$

where the elimination constant \mathbf{Q} is referred to the type $(\Sigma_{x \in A} ([x] =_{A/R} m(d))) / \top$. This term is well typed by effectiveness.

Then by the elimination rule on the quotient type $P/m^*(\mathcal{R})$ it is easy to prove that $Q^{P^{-1}} \cdot Q^P = id$. Moreover, note that $Q^{P^{-1}}$ is mono by the elimination rule on the quotient type and effectiveness. Hence $Q^P \cdot Q^{P^{-1}} = id$ also follows. This concludes the proof that π_1^D is a coequalizer of $(\pi_2^A)^*(\pi_1 \cdot g)$ and $(\pi_2^A)^*(\pi_2 \cdot g)$.

■

5.7 Right adjoints on subobjects

In order to show that in $\mathcal{C}_{\mathcal{T}_{hptop}}$ each pullback functor on subobjects has a right adjoint we first note that the pullback functor on subobjects is isomorphic to the functor $Prop(-) : \mathcal{C}_{\mathcal{T}}^{op} \rightarrow Cat$ defined in the following.

Def. 5.8 For any object $A \in Ob\mathcal{C}_{\mathcal{T}_{hptop}}$, the objects of the category **Prop(A)** are the equivalence classes of mono types depending on A , $B(x) [x \in A]$, under the relation of equiprovability, and the morphisms are the terms $f \in B(x) \rightarrow C(x)$ where $B(x) \rightarrow C(x) \equiv \forall_{B(x)}(C(x))$, since $C(x)$ is mono. The identity is $\lambda y.y \in B(x) \rightarrow B(x)$. The composition of $f \in B(x) \rightarrow C(x)$ and $g \in C'(x) \rightarrow D(x)$, supposing that $C(x)$ is equivalent to $C'(x)$ and in particular that there exists $s \in C(x) \rightarrow C'(x)$, is given by $\lambda y.Ap(g, Ap(s, Ap(f, y))) \in B(x) \rightarrow D(x)$.

Therefore, we can define the above functor $Prop(-) : \mathcal{C}_{\mathcal{T}_{hptop}}^{op} \rightarrow Cat$:

Def. 5.9 For any object $A \in Ob\mathcal{C}_{\mathcal{T}_{hptop}}$, $Prop(A)$ is the above defined category and given a morphism $m \in \mathcal{C}_{\mathcal{T}_{hptop}}(D, A)$ we define $Prop(m)$ as the following functor: for any $B(x) [x \in A]$

$$Prop(m)(B(x) [x \in A]) \equiv B(m(z)) [z \in D]$$

and for every $t \in B(x) \rightarrow C(x)$, given $z \in D$, we define

$$Prop(m)(t) \equiv \lambda w \in B(m(z)).Ap(t[x := m(z)], w)$$

which is a term of type $B(m(z)) \rightarrow C(m(z))$.

We can easily check that $Prop(-)$ is a well defined functor. Then, we recall that the functor $Sub(-) : \mathcal{C}_{\mathcal{T}_{hptop}}^{op} \rightarrow Cat$ is defined as follows: for every $A \in Ob\mathcal{C}_{\mathcal{T}_{hptop}}$, $Sub(A)$ is the poset category $Sub(A)$ of subobjects of $\mathcal{C}_{\mathcal{T}_{hptop}}$, and for every morphism $t : A \rightarrow B$, $Sub(t)$ is the restriction of pullback functor on subobjects. Then we can prove that

Proposition 5.10 *The functor $Sub(-) : \mathcal{C}_{\mathcal{T}_{hptop}}^{op} \rightarrow Cat$ is naturally isomorphic to the functor $Prop(-) : \mathcal{C}_{\mathcal{T}_{hptop}}^{op} \rightarrow Cat$*

and also

Proposition 5.11 *For every morphism $m(y) \in A [y \in D]$ in $\mathcal{C}_{\mathcal{T}_{hptop}}$, there exists the right adjoint of m^* .*

$$Sub(A) \begin{array}{c} \xrightarrow{m^*} \\ \perp \\ \xleftarrow{\forall_m} \end{array} Sub(D)$$

Proof. By the previous proposition, it is enough to show that $Prop(m)$ has a right adjoint. For every mono type $B(y) [y \in D]$ we put

$$\forall_m(B(y) [y \in D]) \equiv \forall_{y \in D}(x =_A m(y)) \rightarrow B(y) [x \in A]$$

whose value at a mono type is indeed a mono type. Then we define a bijection

$$Prop(D)(Prop(m)(C(x)), B(y)) \begin{array}{c} \xrightarrow{\psi_1} \\ \xleftarrow{\psi_2} \end{array} Prop(A)(C(x), \forall_m(B(y)))$$

as follows: for any $t \in C(m(y)) \rightarrow B(y)$ [$y \in D$] we put for any $x \in A$

$$\psi_1(t) \equiv \lambda z. \lambda y. \lambda w. \text{Ap}(t, z)$$

and for any $s \in C(x) \rightarrow \forall_m(B(y))$ [$x \in A$] and any $y \in D$ we put

$$\psi_2(s) \equiv \lambda z. \text{Ap}(\text{Ap}(\text{Ap}(s[x := m(y)]), z), y), \text{eq}_A)$$

It is easy to see that ψ_1 and ψ_2 are inverse to each other and that they are natural on the first variable.

■

5.8 The right adjoint to the pullback functor

In $\mathcal{C}_{\mathcal{T}_{ic}}$ *right adjoint to the pullback functor* is described as in [See84]. For every morphism $m : D \rightarrow A$ of $\mathcal{C}_{\mathcal{T}_{ic}}$, for every object $b : B \rightarrow D$ of $\mathcal{C}_{\mathcal{T}_{ic}}/D$, we put

$$\forall_m(b) \equiv \pi_1 : \Sigma_{x \in A} C(x) \rightarrow A$$

where for $x \in A$

$$C(x) \equiv \forall_{y \in D} (x =_A m(y)) \rightarrow \Sigma_{z \in B} b(z) =_D y$$

5.9 The subobject classifier

In the syntactic category $\mathcal{C}_{\mathcal{T}_{op}}$ the *subobject classifier* is Ω .

The *true* map is

$$\{\top\} \in \Omega [x \in \top]$$

Moreover, given a monomorphism $B \xrightarrow{t} A$ its characteristic map is

$$\{\Sigma_{y \in B} t(y) =_A x\} \in \Omega [x \in A]$$

It is easy to prove that the pullback of the characteristic map with the *True* map is isomorphic to t .

$$\begin{array}{ccc} B & \xrightarrow{\simeq} & \Sigma_{x \in A} \Sigma_{z \in \top} (\{\Sigma_{y \in B} t(y) =_A x\} =_{\Omega} \{\top\}) \\ & \searrow t & \swarrow \pi_1 \\ & & A \end{array}$$

By the equality on Ω and the η -C conversion rule of Ω , the characteristic map is unique.

Indeed, for every $q(x) \in \Omega[x \in A]$ such that

$$\begin{array}{ccc} B & \xrightarrow{\simeq} & \Sigma_{x \in A} \Sigma_{z \in \top} (q(x) =_{\Omega} \{\top\}) \\ & \searrow t & \swarrow \pi_1 \\ & & A \end{array}$$

by η -C conversion rule of Ω and by the equality on Ω

$$q(x) = \{\text{Eq}(\Omega, q(x), \{\top\})\} = \{\Sigma_{y \in B} t(y) =_A x\}$$

■

6 The categorical semantics

The idea of the semantics that goes back to [See84] is to interpret the calculus using the codomain fibration [Jac99].

Our notion of model for the dependent typed calculi described in the previous section combines the notion of categorical semantics based on display maps [See84, HP89], together with the tools provided

by contextual categories to interpret substitution correctly [Car86]. This is the same notion used in [Mai98b] to provide the semantics for the typed calculi of Heyting pretopoi and topoi.

Our models correspond to particular contextual categories, namely only those arising from the split fibration associated to the codomain fibration (see [Jac99] for definitions), where the category of contexts is equivalent to the category under consideration. The use of split fibrations is required to interpret substitution correctly, since the natural use of the codomain fibration as in [See84] gives rise to coherence problems first solved by [Hof94].

While the validity and completeness theorem is straightforward for contextual categories in general, the same theorem with respect to our contextual categories requires more care. However, we want to point out that it holds, hence *it is sufficient to use the split fibration associated to the codomain fibration to prove validity and completeness theorems between a dependent type theory and corresponding categories and this is necessary to see that the calculi even provide the internal languages of the corresponding categories modelling them.*

Now to explain in more detail, suppose to want to interpret the dependent typed calculus \mathcal{T} in the category \mathcal{C} , which is at least lex. Then, the idea is to interpret the judgement $B[\Gamma]$ as a suitable sequence of morphisms of \mathcal{C} to the terminal object 1 and the judgement $b \in B[\Gamma]$ as a section of the last morphism of the sequence representing the dependent type B .

To this purpose we define the following category of paths of \mathcal{C} :

Def. 6.1 *Given a category \mathcal{C} with terminal object 1 , the objects of the category $\mathbf{Pgr}(\mathcal{C})$ are finite sequences a_1, a_2, \dots, a_n of morphisms of \mathcal{C}*

$$A_n \xrightarrow{a_n} \dots A_2 \xrightarrow{a_2} A_1 \xrightarrow{a_1} 1$$

and a morphism from a_1, a_2, \dots, a_n to b_1, b_2, \dots, b_m is a morphism b of \mathcal{C} such that $b_n \cdot b = a_n$

$$\begin{array}{ccc} & & b \\ & & \searrow \\ A_n & \xrightarrow{\quad} & B_n \\ & \searrow a_n & \swarrow \\ & & \\ 1 \xleftarrow{!_{A_1}} A_1 & \dots & \xleftarrow{a_{n-1}} A_{n-1} \\ & & \swarrow b_n \end{array}$$

provided $n = m$ and $a_i = b_i$ for $i = 1, \dots, n - 1$.

However, since we would like to interpret substitution by means of pullback and the natural choice of the reindexing pullback pseudofunctor to interpret substitution brings coherence problems (due to the fact this is *not* a functor), we need to use fibred functors, as in [Hof94], to interpret substitution correctly. In other categorical terms this means to pass from the codomain fibration to its split one. To this purpose we define a category of paths of fibred functors corresponding to $\mathbf{Pgr}(\mathcal{C})$.

Before doing it, we recall that a fibred functor $\sigma : \mathbf{ObFib}(\mathcal{C}/A, \mathcal{C}^\rightarrow)$ is a functor $\sigma : \mathcal{C}/A \rightarrow \mathcal{C}^\rightarrow$ which is fibred (see [Jac99]) from the fibration $\mathit{dom}_{\mathcal{C}}$ to the fibration $\mathit{cod}_{\mathcal{C}}$, i.e. it sends cartesian morphisms of $\mathit{dom}_{\mathcal{C}}$ to cartesian morphisms of $\mathit{cod}_{\mathcal{C}}$.

Def. 6.2 *Given a lex category \mathcal{C} , the objects of the category $\mathbf{Pgf}(\mathcal{C})$ are finite sequences $\sigma_1, \sigma_2, \dots, \sigma_n$ of fibred functors $\sigma_i : \mathbf{ObFib}(\mathcal{C}/A_i, \mathcal{C}^\rightarrow)$ for $i = 1, \dots, n$ such that $\sigma_1(\mathit{id}_{A_1}), \sigma_2(\mathit{id}_{A_2}), \dots, \sigma_n(\mathit{id}_{A_n})$ is an object of $\mathbf{Pgr}(\mathcal{C})$. The morphisms of $\mathbf{Pgf}(\mathcal{C})$ from $\sigma_1, \sigma_2, \dots, \sigma_n$ to $\tau_1, \tau_2, \dots, \tau_m$ are defined only if $n = m$ and $\sigma_i = \tau_i$ for $i = 1, \dots, n - 1$ and $\sigma_n, \tau_n \in \mathbf{ObFib}(\mathcal{C}/A_n, \mathcal{C}^\rightarrow)$ and they are natural transformations from the functor σ_n to τ_n such that for every $b : B \rightarrow A_n$ the second member of $\rho(b)$ is the identity (recall that $\rho(b)$ is a morphism of \mathcal{C}^\rightarrow), that is the triangle*

$$\begin{array}{ccc} & \xrightarrow{\rho_1(b)} & \\ & \searrow & \swarrow \\ \sigma_n(b) & & \tau_n(b) \\ & \searrow & \swarrow \\ & B & \end{array} \quad \text{commutes.}$$

The use of fibred functors to interpret substitution correctly in some sense forces us to define a preinterpretation

$$\tilde{\mathcal{I}}_{\mathcal{C}} : \mathcal{T} \longrightarrow \mathbf{Pgf}(\mathcal{C})$$

on the type and term judgements derivable in the typed calculus \mathcal{T} towards the paths of fibred functors. The preinterpretation essentially says how to interpret a dependent type and a typed term after any

possible substitution. Then the interpretation of type and term judgements corresponds to evaluate their preinterpretations on the identical substitution.

Hence, we define a valuation $\mathcal{V} : Pgf(\mathcal{C}) \longrightarrow Pgr(\mathcal{C})$ in this manner: for every object of $Pgf(\mathcal{C})$ $\sigma_1, \sigma_2, \dots, \sigma_n$

$$\mathcal{V}(\sigma_1, \sigma_2, \dots, \sigma_n) \equiv \sigma_1(id_{A_1}), \sigma_2(id_{A_2}), \dots, \sigma_n(id_{A_n})$$

where $A_i = I(\sigma_i)$ for $i = 1, \dots, n$, and for every morphism ρ of $Pgf(\mathcal{C})$ between $\sigma_1, \sigma_2, \dots, \sigma_n$ and $\tau_1, \tau_2, \dots, \tau_n$

$$\mathcal{V}(\rho) \equiv \rho(id_{A_n})$$

and finally the interpretation $\mathcal{I}_\mathcal{C} : \mathcal{T} \longrightarrow Pgr(\mathcal{C})$ is defined as $\mathcal{I}_\mathcal{C} \equiv \mathcal{V} \cdot \tilde{\mathcal{I}}_\mathcal{C}$

$$\begin{array}{ccc} Au & \xrightarrow{\mathcal{I}_\mathcal{C}} & Pgr(\mathcal{C}) \\ & \searrow \tilde{\mathcal{I}}_\mathcal{C} & \nearrow \mathcal{V} \\ & Pgf(\mathcal{C}) & \end{array}$$

More precisely, a *dependent type* judgement of \mathcal{T}

$$B(x_1, \dots, x_n) [x_1 \in A_1, \dots, x_n \in A_{n-1}(x_1, \dots, x_{n-1})]$$

will be preinterpreted as the object of $Pgf(\mathcal{C})$

$$\alpha_1, \alpha_2, \dots, \alpha_n, \beta$$

and then interpreted as

$$1 \xleftarrow{\alpha_1(id)} A_{\Sigma_1} \dots \xleftarrow{\alpha_n(id)} A_{\Sigma_n} \xleftarrow{\beta(id)} B_\Sigma$$

The *equality between types* will be preinterpreted as equality between objects of $Pgf(\mathcal{C})$ and hence interpreted as the equality between objects of $Pgr(\mathcal{C})$.

At this point we want to remark that to correctly interpret equality between types as equality between functors we assume that this equality between functors is well defined and hence, it follows that equality between categorical objects has to be defined, too. A simple way out would be to assume that all our categories are small.

A term judgement of \mathcal{T}

$$b \in B(x_1, \dots, x_n) [\Gamma_n]$$

will be preinterpreted as a natural transformation b^I from $\alpha_1, \alpha_2, \dots, \alpha_n, i_{A_{\Sigma_n}}$ to $\alpha_1, \alpha_2, \dots, \alpha_n, \beta$, and then interpreted as $b^I(id)$, that is a section of $\beta(id)$

$$\begin{array}{ccc} A_{\Sigma_n} & \xrightarrow{b(id)} & B_\Sigma \\ & \searrow id & \nearrow \beta(id) \\ 1 \xleftarrow{!_{A_{\Sigma_1}}} A_{\Sigma_1} \dots \xleftarrow{\alpha_n(id)} A_{\Sigma_n} & & \end{array}$$

provided that the type judgement $B(x_1, \dots, x_n) [\Gamma_n]$ is interpreted as $1 \xleftarrow{\alpha_1(id)} A_{\Sigma_1} \dots \xleftarrow{\alpha_n(id)} A_{\Sigma_n} \xleftarrow{\beta(id)} B_\Sigma$.

The *equality between terms* will be preinterpreted as equality between natural transformations and hence interpreted as the equality between morphisms of $Pgr(\mathcal{C})$.

To be short, we do not proceed in giving the definition of the interpretation of the various dependent typed calculi of section 3 in the corresponding categories. However, the interpretation follows the correspondence in the table in section 4 (see for example [Mai98b] or [Mai98a] for details on the interpretation of some of the calculi presented here and proofs of the corresponding validity and completeness theorems).

However, we point out that a mono type $B(x) [x \in A]$ will turn out to be interpreted in a sequence of morphisms whose last one $\beta(id)$ is a monomorphism, thanks to the fact that the valid interpretation of the judgement

$$y = z \in B(x) [x \in A, y \in B(x), z \in B(x)]$$

says that the kernel pair of $\beta(id)$ is the identity relation.

Now, we just state the validity and completeness for *lex* categories with respect to \mathcal{T}_{lex} :

Theorem 6.3 (Validity and completeness) *If A type $[\Gamma_n]$ is derivable in \mathcal{T}_{lex} then $\mathcal{I}_C(A \text{ type } [\Gamma_n])$ is well defined for any lex category \mathcal{C} . If $a \in A [\Gamma_n]$ is derivable in \mathcal{T}_{lex} then $\mathcal{I}_C(a \in A [\Gamma_n])$ is well defined for any lex category \mathcal{C} .*

Suppose that A type $[\Gamma_n]$ and B type $[\Gamma_n]$ are derivable in \mathcal{T}_{lex} , then $A = B [\Gamma_n]$ is derivable in \mathcal{T}_{lex} iff we have $\mathcal{I}_C(A \text{ type } [\Gamma_n]) = \mathcal{I}_C(B \text{ type } [\Gamma_n])$ for every lex category \mathcal{C} .

Suppose that $a \in A [\Gamma_n]$ and $b \in A [\Gamma_n]$ are derivable in \mathcal{T}_{lex} then $a = b \in A [\Gamma_n]$ is derivable in \mathcal{T}_{lex} iff we have $\mathcal{I}_C(a \in A [\Gamma_n]) = \mathcal{I}_C(b \in A [\Gamma_n])$ for every lex category \mathcal{C} .

The same validity and completeness theorem holds

- for \mathcal{T}_{alex} with respect to *arithmetic lex categories*,
- for \mathcal{T}_{reg} with respect to *regular categories*,
- for \mathcal{T}_{dis} with respect to *distributive categories*,
- for \mathcal{T}_{loc} with respect to *locoi*,
- for \mathcal{T}_{ptop} with respect to *pretopoi*,
- for \mathcal{T}_{hptop} with respect to *Heyting pretopoi*,
- for \mathcal{T}_{au} with respect to *arithmetic universes*,
- for \mathcal{T}_{lcc} with respect to *locally cartesian closed categories*,
- for \mathcal{T}_{top} with respect to *topoi*.

7 Dependent type theory as internal language.

Given any of the considered category in section 2, for example a lex category \mathcal{C} , we can describe its internal dependent type theory as a theory $T(\mathcal{C})$ of the corresponding calculus it models, in this case \mathcal{T}_{lex} . We recall that a validity and completeness theorem of a calculus \mathcal{T} with respect to lex categories is not in general enough to guarantee that the internal language of a lex category is a theory of \mathcal{T} (where with *theory* we mean an extension of the calculus \mathcal{T} with axioms). However, in the correspondence so far described the type theoretic constructors provide the internal language for the categorical properties in the table of section 4. The proof of this just relies on the fact that completeness holds even with respect to models described only via the codomain fibration as seen above.

Here, we just briefly mention the theorems about the internal language without many details, taking lex categories and \mathcal{T}_{lex} as a leading sample. The same can be stated and proved also for the other calculi with respect to the corresponding categories modelling them, following the same technique developed in [Mai98a] for Heyting pretopoi (see also [Mai98b]).

The internal type theory of a lex category \mathcal{C} is based on the initial type theory \mathcal{T}_{lex} augmented with the specific type and term judgements of \mathcal{C} . As in the interpretation of a typed calculus in the categorical semantics adopted here, the idea is that a type judgement corresponds to an object of $Pgr(\mathcal{C})$ obtained as the evaluation on the identical substitution of an object of $Pgf(\mathcal{C})$, which represents a dependent type with all its possible substitutions. Analogously a term judgement corresponds to a morphism of $Pgr(\mathcal{C})$ obtained as the evaluation on the identical substitution of a morphism of $Pgf(\mathcal{C})$, which represents a dependent term with all its possible substitutions.

To be clearer, for any sequence of fibred functors $\alpha_1, \alpha_2, \dots, \alpha_n, \beta$ of $Pgf(\mathcal{C})$, we define

$$\beta^{-1}(x_1, \dots, x_n)[x_1 \in \alpha_1^{-1}, \dots, x_n \in \alpha_n^{-1}(x_1, \dots, x_{n-1})]$$

as the type judgement corresponding to

$$B \xrightarrow{\beta(id)} A_n \xrightarrow{\alpha_n(id)} \dots \xrightarrow{\alpha_1(id)} 1$$

- \mathcal{T}_{hptop} provides the internal dependent type theory for *Heyting pretopoi*,
- \mathcal{T}_{au} provides the internal dependent type theory for *arithmetic universes*,
- \mathcal{T}_{lcc} provides the internal dependent type theory for *locally cartesian closed* categories,
- \mathcal{T}_{top} provides the internal dependent type theory for *topoi*.

We can establish an analogous correspondence “internal type theory/category” also between a calculus obtained as combination of the type constructors in the table of section 4 with respect to the categories enjoying the corresponding combination of categorical properties. Of course the correspondence “internal type theory/category” implies that of “type theory/category as model” via a validity and completeness theorem as shown in this section.

8 Conclusions

In the future work we intend to proceed with applying the correspondence between categories and dependent type theories so far obtained:

- to reason within the categories by using logical proofs,
- viceversa, to import in type theory categorical proofs.

Acknowledgements. I wish to thank the organizers and guests met during my staying in May/June 2001 at the Mittag-Leffler Institute for the fruitful period spent there, and among them especially Per Martin-Löf, Giovanni Sambin, Peter Aczel, Steve Awodey, Andrej Brauer and Erik Palmgren for many interesting discussions about this research topic. Finally, I wish to thank Silvio Valentini for his continuous support on this work.

References

- [AB01] S. Awodey and A. Bauer. Propositions as [types]. Technical report, Institut Mittag-Leffler, n.34, June 2001.
- [Bel88] J.L. Bell. *Toposes and Local Set Theories: an introduction*. Clarendon Press, Oxford, 1988.
- [Ben85] J. Benabou. Fibred categories and the foundations of naive category theory. *Journal of Symbolic Logic*, 50:10–37, 1985.
- [Car86] J. Cartmell. Generalised algebraic theories and contextual categories. *Annals of Pure and Applied Logic*, 32:209–243, 1986.
- [CG88] T. Coquand and G.Huet. The calculus of constructions. *Information and Computation*, 76:95–120, 1988.
- [Coc90] J.R.B. Cockett. List-arithmetic distributive categories: locoi. *Journal of Pure and Applied Algebra*, 66:1–29, 1990.
- [Col73] J. C. Cole. Categories of sets and models of set theory. In *Proc. B. Russell Memorial Logic Conf.*, Leeds, 1973. J. Bell and A. Slomson.
- [Con86] R. Constable et al. *Implementing mathematics with the Nuprl Development System*. Prentice Hall, 1986.
- [dB91] N.G. de Bruijn. Telescopic mapping in typed lambda calculus. *Information and Computation*, 91:189–204, 1991.
- [Dia75] R. Diaconescu. Axiom of choice and complementation. *Proc. Amer. Math. Soc.*, 51:176–178, 1975.

- [Hof94] M. Hofmann. On the interpretation of type theory in locally cartesian closed categories. In Proceedings of CSL'94, September 1994.
- [Hof95] M. Hofmann. *Extensional concept in intensional type theory*. PhD thesis, University of Edinburgh, July 1995.
- [HP89] J.M.E. Hyland and A. M. Pitts. The theory of constructions: Categorical semantics and topos theoretic models. In J. W. Gray and A. Scedrov, editors, *Categories in Computer Science and Logic*, volume 92 of *Contemporary Mathematics*, pages 137–199, 1989.
- [Jac99] B. Jacobs. *Categorical Logic and Type Theory*., volume 141 of *Studies in Logic*. Elsevier, 1999.
- [JM95] A. Joyal and I. Moerdijk. *Algebraic set theory*., volume 220 of *Lecture Note Series*. Cambridge University Press, 1995.
- [Joh77] P. Johnstone. *Topos theory*. Academic Press, 1977.
- [LS86] J. Lambek and P. J. Scott. *An introduction to higher order categorical logic*., volume 7 of *Studies in Advanced Mathematics*. Cambridge University Press, 1986.
- [Mac71] S. Mac Lane. *Categories for the working mathematician*., volume 5 of *Graduate text in Mathematics*. Springer, 1971.
- [Mai98a] M.E. Maietti. The internal type theory of an Heyting Pretopos. In C.Paulin-Mohring E. Gimenez, editor, *Types for Proofs and Programs. Selected papers of International Workshop Types '96, Aussois*, volume 1512 of *LNCS*, pages 216–235. Springer Verlag, 1998.
- [Mai98b] M.E. Maietti. *The type theory of categorical universes*. PhD thesis, University of Padova, February 1998.
- [Mai99a] M.E. Maietti. About effective quotients in constructive type theory. In W. Naraschewski T. Altenkirch and B. Reus, editors, *Types for proofs and programs. International workshop, TYPES '98. Kloster Irsee, Germany, March 27-31. 1999*, volume 1657 of *Lectures Notes in Computer Science*, pages 164–178. Springer Verlag, 1999.
- [Mai99b] M.E. Maietti. The typed calculus of arithmetic universes. Technical report, University of Birmingham, CSR-99-14, December 1999. also Technical Report- University of Padova n.5 Dec. 1999.
- [Mar75] P. Martin-Löf. An intuitionistic theory of types: predicative part. In J. Smith G. Sambin, editor, *Logic Colloquium '73 (Bristol, 1973)*, volume 80 of *Studies in Logic and the Foundations of Mathematics*, pages 73–118. North-Holland, Amsterdam, 1975.
- [Mar84] P. Martin-Löf. *Intuitionistic Type Theory, notes by G. Sambin of a series of lectures given in Padua, June 1980*. Bibliopolis, Naples, 1984.
- [Mar95] P. Martin-Löf. An intuitionistic theory of types. In J. Smith G. Sambin, editor, *Twenty five years of Constructive Type Theory*, pages 127–172. Oxford Science Publications, 1995.
- [Mit72] W. Mitchell. Boolean topoi and the theory of sets. *J. Pure Appl. Alg.*, 2:261–274, 1972.
- [MM92] S. MacLane and I. Moerdijk. *Sheaves in Geometry and Logic. A first introduction to Topos theory*. Springer Verlag, 1992.
- [MR77] M. Makkai and G. Reyes. *First order categorical logic*., volume 611 of *Lecture Notes in Mathematics*. Springer Verlag, 1977.
- [MV99] M.E. Maietti and S. Valentini. Can you add powersets to Martin-Löf intuitionistic type theory? *Mathematical Logic Quarterly*, 45:521–532, 1999.

- [NPS90] B. Nordström, K. Peterson, and J. Smith. *Programming in Martin Löf's Type Theory*. Clarendon Press, Oxford, 1990.
- [Pit95] A.M. Pitts. Categorical logic. In Oxford University Press, editor, *Logical Methods in Computer Science*, volume VI of *Handbook of Logic in Computer Science*, page To appear, 1995.
- [See84] R. Seely. Locally cartesian closed categories and type theory. *Math. Proc. Cambr. Phyl. Soc.*, 95:33–48, 1984.
- [Smi88] J. Smith. The independence of Peano's fourth axiom from Martin Löf's type theory without universes. *Journal of Symbolic Logic*, 53, 1988.
- [Str91] Th. Streicher. *Semantics of type theory*. Birkhäuser, 1991.
- [Swa91] M. D. G. Swaen. The logic of first order intuitionistic type theory with weak sigma-elimination. *J. Symbolic Logic*, 56:467–483, 1991.
- [Swa92] M. D. G. Swaen. A characterization of ML in many-sorted arithmetic with conditional application. *J. Symbolic Logic*, 57:924–953, 1992.
- [Tay97] P. Taylor. *Practical Foundations of Mathematics*, volume 99 of *Cambridge studies in advanced mathematics*. Cambridge University Press, 1997.
- [Wra85] G. C. Wraith. Notes on arithmetic universes and Gödel incompleteness theorems. Unpublished manuscript., 1985.